A FRAMEWORK FOR THE ANALYSIS AND COMPARISON OF PROCESS MINING ALGORITHMS

by

PHILIP WEBER

A thesis submitted to the
University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
July 2013

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

Abstract

Process mining algorithms use event logs to learn and reason about business processes. Although process mining is essentially a machine learning task, little work has been done on systematically analysing algorithms to understand their fundamental properties, such as how much data is needed for confidence in mining. Nor does any rigorous basis exist on which to choose between algorithms and representations, or compare results.

We propose a framework for analysing process mining algorithms. Processes are viewed as distributions over traces of activities and mining algorithms as learning these distributions. We use probabilistic automata as a unifying representation to which other representation languages can be converted.

To validate the theory we present analyses of the Alpha and Heuristics Miner algorithms under the framework, and two practical applications. We propose a model of noise in process mining and extend the framework to mining from 'noisy' event logs. From the probabilities and sub-structures in a model, bounds can be given for the amount of data needed for mining. We also consider mining in non-stationary environments, and a method for recovery of the sequence of changed models over time.

We conclude by critically evaluating this framework and suggesting directions for future research.

This thesis is dedicated to my wife, Louise,
and daughters, Bethan, Ruth and Anna,
without whom it would not exist.

ACKNOWLEDGEMENTS

First and foremost, my sincerest thanks to Dr. Behzad Bordbar and Dr. Peter Tiňo who supervised this work. I am amazed at your patience as I fumblingly started out in research. Thank you for guiding and constantly challenging me, and for holding me to intellectual and mathematical rigour. Above all, for helping me learn to think and write.

My road from industry to academia started with redundancy. Thus I owe thanks to my former employer, but more importantly to former colleagues who showed interest in my ideas and provided practical support. Thanks also to members of the School of Computer Science, University of Birmingham, who believed I could succeed in a research Masters, and to fellow students for friendship, support and intellectual stimulation.

My 'thesis group' members at Birmingham helped me develop my research ideas. Thank you for your interest and taking the time to read reports, attend meetings, ask probing questions and keep me on the road to achieving a thesis. I also thank the undergraduate students who unwittingly helped me to express myself more clearly.

Basim Majeed and colleagues at the Etisalat BT Innovation Centre (EBTIC) invited me to spend three months working on an industrial process mining project in Abu Dhabi. Thank you for this priceless opportunity to consider my ideas in a practical setting, not to mention, to experience another part of the world.

Without the ProM Process Mining Framework and other open source tools, process mining research would be much harder to put into practice. Thanks to all the developers for offering this work to the community. Thanks also to all the anonymous reviewers of my academic paper submissions. Your patience, generosity with your time and helpful comments have helped me become a member of the academic community.

This work was supported by a Doctoral Training Grant funded by EPSRC and the School of Computer Science, University of Birmingham. Without this support, it could not have gone ahead.

Last but definitely not least, I owe a boundless debt of gratitude to my family for allowing me this opportunity to study, putting up with the times I found it difficult, making do without me when I was absent, and tolerating my increased presence at other times. Without you there is absolutely no chance I could have completed this thesis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION

$\mathcal{A}$	A set of business activities.
$\mathcal{M}$	'Ground truth' model (may be unknown).
P_M	Probability distribution over traces represented by model $\mathcal{M}$
Σ	Alphabet of symbols encoding business activities.
$\{a, b, \ldots\} \in \Sigma$	Valid business activities in the process.
$\{x, y, \ldots\} \in \Sigma^+$	Non-empty strings representing sequences of activities.
$\mathcal{T}$	The set of all valid process traces (cases).
xy	The concatenation of strings x and y.
$x\Sigma^*$	The set of strings with x as prefix.
$\Sigma^* x$	The set of strings with x as suffix.
$\Sigma^* x \Sigma^*$	The set of strings with x as sub-string.
$\pi(x)$	Probability of sub-string ab occurring in a trace.
$\pi(\rightarrow a)$	Probability of 'reaching' a, i.e. $\pi(\rightarrow a) = \pi(i\Sigma^* a \Sigma^* o)$.
$\mathcal{W} \subset \{x \vert x \in \mathcal{T}\}$	Event or Workflow log, a *bag* or *multi-set* of traces.
$\mathcal{W}_n$	Event log of n traces.
$N(x)$	Frequency of sub-string x in $\mathcal{W}$.
$Q_n\big(N(x)\big)$	Distribution of $N(x)$ in event log of n traces.
Q_A, q	Set of states in PDFA A, single state.
$q_0, q_F \in Q_A$	Single start, end state in PDFA A.
δ_A	Conditional transition probability function between states in PDFA A (arc probabilities).
$d_2(P_1, P_2)$	Euclidean Distance between probability distributions P_1, P_2.
$d_{Bhat}(P_1, P_2)$	Bhattacharyya Distance between P_1, P_2.
$d_{KL}(P_1, P_2)$	Kullback-Leibler Divergence between P_1, P_2.
$d_{JSD}(P_1, P_2)$	Jensen-Shannon Divergence between P_1, P_2.
$a \rightarrow b$	Arc representing causal dependency from a to b.
$a \rightarrow (b_1 \parallel b_2 \ldots)$	Parallel ('AND') split, from a to paths starting $b_1, b_2, \ldots$.
$a \rightarrow (b_1 \# b_2 \ldots)$	'XOR' split from a to alternative paths starting $b_1, b_2, \ldots$.
DM_{ab}	Heuristics Miner Dependency Measure between a and b.
$\mathrm{DM}_{ab,n}$	Dependency Measure calculated from event log of n traces.
$\gamma_n(E)$	Probability of a complex event E (specified in context).
$P_{\alpha,n}(S)$	Probability that the Alpha Algorithm mines structure S correctly from the event log.
$P_{HM,n}(S)$	*ditto* Heuristics Miner.
$c\big(N(ia), N(ib)\big)$	Correlation between number of sub-strings ia, ib (Chapter 6).

ABBREVIATIONS

ACM	Adaptive Case Management
BAM	Business Activity Monitoring
BI	Business Intelligence
BPA	Business Process Analysis
BPEL	Business Process Execution Language
BPG	Business Process Graph
BPM	Business Process Management
BPMN	Business Process Modelling Notation
CRM	Customer Relationship Management
DM	Dependency Measure (HM)
DT	Dependency Threshold (HM)
EPC	Event-driven Process Chain
ERP	Enterprise Resource Planning
HM	Heuristics Miner
HMM	Hidden Markov Model
JSD	Jensen-Shannon Divergence
KL	Kullback-Leibler (Divergence)
OMG	Object Management Group
PAC	Probably Approximately Correct
PDFA	Probabilistic Deterministic Finite Automaton
PM	Process Mining
PN	Petri Net
PO	Positive Observations (HM)
RG	Reachability Graph
RTB	Relative To Best (HM)
RTBPM	Real-Time Business Process Mining
SWF-Net	Sound Workflow Net
UH	'Use All-Activities-Connected' Heuristic (HM)
UML	Unified Modelling Language
WF	Workflow
XOR	Exclusive Or
YAWL	Yet Another Workflow Language

Part I

Framework for the Analysis of Process Mining Algorithms

CHAPTER 1

INTRODUCTION

In this thesis we present, demonstrate and evaluate a framework within which to objectively analyse and compare process mining algorithms. We show that this provides a rigorous foundation for addressing important process mining questions, such as assessing the quality of process models, comparing results from different algorithms, dealing with changing processes, or mining from noisy data. In this chapter we set this work in context. We introduce process mining and establish the need for the work presented in this thesis.

1.1 Interconnected and Regulated Business World

Defining characteristics of the modern world are pressures on time and resources, and the ubiquitousness of computer systems. Businesses operate in increasingly competitive markets, with reducing margins and increasing costs. Globalisation of business and communications has resulted in on the one hand, huge and complex multinational corporations, and on the other, complex international interaction between businesses. Information systems and telecommunications underpin and facilitate this intra- and inter-business complexity.

Managing, tracking and securing business activity therefore becomes increasingly difficult. Notorious frauds (e.g. Enron, Worldcom), and the financial crises beginning from 2008 have raised awareness of the complexity of banking systems; food distribution scandals have created public awareness of the complex processes bringing food products from

supplier to consumer; and public transport and social and health services failures have highlighted the complex interactions between public bodies delivering government services. Widely rumoured industrial and state-sponsored 'cyber-attacks' (e.g. [31, 61, 105]) have increased government and industry awareness of security risks inherent in reliance on information systems and internet-based services (see e.g. [58]), especially when their operation and interaction is not fully understood.

These risks and crises have led to increasing business regulation: financial regulations such as the Sarbanes-Oxley Act and Basel Accords; cyber-security regulation including the US Homeland Security act; regulatory bodies such as the UK Financial Services Authority and Information Commissioner's Office. All impose requirements for organisations to show that they manage their operations according to certain standards.

Businesses are thus under pressure to understand and manage their operations, in order to cut down waste and optimise efficiency, show adherence to regulation, and maintain their public image. This has engendered increasing interest in defining, understanding and managing business processes, to specify and enforce how the business operates.

1.2 Processes and Process Management

A business process is defined as 'step-by-step activities to solve a business problem or need' [76]. It runs continuously, but may be inactive until triggered by an event. For example, receiving a customer order causes the order to be recorded and routed to the correct department. The process may then become inactive again until order processing begins. Typical business processes include fulfilling customer orders, paying invoices, making purchases, changing IT systems, handling customer complaints, and so on.

Figure 1.1 is an informal view of a simplified process for fulfilling an customer order. We will use this as a running example throughout this thesis. The supplier receives an order (1), e.g. via its website or from another business, and registers it on an order processing system ('open order'). Next, stock is checked (2), which involves liaison between

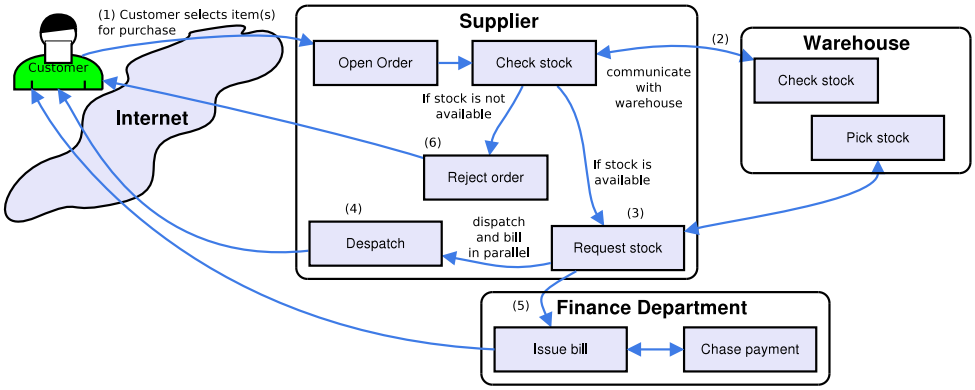


Figure 1.1: Informal Business Process Diagram for fulfilling a Customer Order

the supplier and a warehouse department or business. If adequate stock is found (3), the items are requested and despatched (4) and, in parallel, the bill issued by the finance department. If stock is not available, the order is cancelled (6). The finance department ensures payment (5). The process may also describe details such as who should carry out the activities, timing, rules for following particular sequences of activities.

Reasons to analyse such a process include ensuring the order is satisfied correctly and quickly, optimising the availability of stock, or ensuring prompt payment. Methodologies such as Business Process Management (BPM), Business Activity Monitoring (BAM) and Business Process Analysis (BPA) [155, 158], loosely grouped under the term 'Business Intelligence', have been developed to manage and analyse this kind of structured process. Formal languages such as Petri nets [112], BPMN [109] or BPEL [88] allow models of processes to be rigorously defined and analysed. Recently, Adaptive Case Management (ACM) [140] has been introduced to manage less structured work.

These methodologies and tools require process models to be manually created by business analysts and management. Since these rely on human understanding of the process, the resultant models may be incomplete or reflect desired operation rather than reality. Business Intelligence tools provide metrics for performance and costs, etc,, but provide no deep insight into the process behaviour such as interactions between activities, people, and organisations; what governs different paths through the process, or whether audit requirements are met. Process mining seeks to address these problems.

1.3 Process Mining

Process mining [144,148,155] is the discovery and analysis of models of business processes, from data recorded in 'event logs' by business information systems. It brings together ideas and techniques from business process modelling, data mining and machine learning.

Data mining involves finding and making use of complex and novel patterns and knowledge in data [56], while machine learning (e.g. [103,104]) is concerned with computers using data to learn to perform complex tasks and review and improve their performance. This includes building models of data, for example to understand its structure or allow new data to be classified. Process mining involves both modelling the structure of the process (from data) and discovering patterns to give insight into the process behaviour.

A business process can be considered from various 'perspectives' [155]. We focus on the control-flow, understanding what activities take place and in what order. This is broadly split into *discovery*, *conformance* and *extension*.

Process discovery algorithms aim to reconstruct the underlying business process structure based on a sample event log containing a record of the enactment of process activities. Algorithms typically assume that for each event is recorded as a minimum an activity name, the case (single run through the process) to which it belongs, and either a start or end time, or that activities are recorded in order of occurrence. Data is assumed to be in a suitable format for mining, via a pre-processing step. MXML [167] and XES [178] are standard event log formats used by many algorithms. Activity names are normally assumed to be unique. Some algorithms relax some of these assumptions, for example using both start and end times to obtain activity durations [29,117], allowing non-unique activity names [78–80], or not requiring case IDs [63].

Many algorithms have been proposed, with different theoretical foundations and biases, e.g. [20, 46, 63, 67, 133, 145, 156, 164, 164, 194, 195] (Chapter 3). This is a difficult problem. Gold [69] showed that learning a regular grammar from example sequences of symbols is NP-hard. Although event logs also record activities sequentially, they may have occurred in parallel (e.g. carried out concurrently by different people). As the number of such

parallel activities increases, the number of possible ways in which they may be ordered increases exponentially [155]. Discovery algorithms must differentiate between alternate and parallel behaviour; which becomes additionally difficult when some sequences have low probability of occurring.

Furthermore, algorithms usually produce process models for human interpretation. Therefore processes are often assumed to be structured [133, 154], e.g. with matching splits and joins, and limits are often required on the visual complexity of the mined model. Business process modelling languages may use elements such as 'hidden activities' to define splits and joins, for which there is no direct evidence in the event log and which must be inferred by the mining algorithm. Finally, problems in recording the event log or errors in following the process may lead to discrepancies between the underlying process and the record in the event log. This is sometimes interpreted as 'noise'.

Most process discovery algorithms capture process control flow using non-probabilistic representations in which nodes describe activities or groups of activities. These range from simple directed graphs [7,44], to process-specific representations (e.g. [70,133]) which support concurrency and process structures [154], to Petri nets [112, 157], the most common representation used by algorithms mining structured processes (e.g. [20, 156, 195]). Less formal representations [73, 172, 194] have been introduced to represent flexible processes.

Conformance analysis [12, 122, 124] is concerned with assessing the quality of process mining results and comparing models, while process *extension* deals with using mined models to enhance or derive additional information about the process, e.g. to optimise [150, 172], understand decisions [128], predict outcomes of active processes [131, 135, 165], or simulate changes [123, 125].

Process mining is an active area of research and new algorithms continue to be developed. Important open questions include better mining of processes with concurrent and cyclical behaviour, and the 'completeness' of event logs: how much data is needed to ensure sufficient representative process behaviour for the algorithm to mine it correctly [156]? Other challenges are mining unstructured processes, event logs recorded at different levels

of abstraction or containing events from multiple processes, or affected by 'noise' so as to not faithfully represent the underlying process; balancing representing the observed behaviour with generalising to unseen data; dealing with non-stationary processes. Many of these challenges are summarised in the IEEE Process Mining Manifesto [149].

1.4 The Lack of a Foundation for Process Mining

Many of these questions are difficult to answer objectively, because there is no agreed framework within which to investigate them. Likewise, there is no formal basis against which to evaluate and compare process models and mining algorithms. How should process models in different representations be compared? Many existing conformance measures are tied to specific representations. Is it reasonable to compare models from algorithms with fundamentally different aims and biases? What basis do we have for measuring the completeness of an event log, to know whether or not a process has changed, or to generalise a model to unseen data while faithfully representing observed behaviour?

Because of the diversity of algorithms and representations, and these currently unanswerable questions, methods are needed for analysing the behaviour of algorithms. The review of process mining in [144], and the comparison of metrics in [122] come to the same conclusion: that more research is required into developing generic frameworks for considering such process mining questions. This thesis introduces and analyses one such framework to provide a common basis for analysing process mining algorithms and models, independent of any representation.

1.5 Structure of this Thesis

In Chapter 2 we review business process modelling from a process mining perspective. We focus on the most common process representations used by process mining algorithms, Petri nets, Causal Matrices and Heuristics Nets, and review other representations. We

introduce probabilistic automata (PDFA) as a 'common denominator' to which other process representations can be converted, and relate them to Petri nets. PDFA underpin our framework for considering process mining algorithms.

In Chapter 3 we introduce business process mining and review process mining algorithms and process metrics. We describe two common algorithms in more depth, the Alpha Algorithm [156] and Heuristics Miner [194], which are analysed in later chapters under the framework presented in this thesis. We introduce and motivate the problem addressed by the thesis, namely the lack of a common basis to discuss and compare process mining algorithms and results, and to investigate the behaviour of process mining algorithms. We discuss two problems: firstly, that the prevailing view of business processes and process mining does not take adequate account of the stochastic nature of processes; and secondly, the lack of methods for comparing the many algorithms, representations, and metrics for assessing models.

Chapter 4 is the core of the thesis, presenting the theory of our framework for the analysis of process mining algorithms. We propose a machine learning view of process mining, in which business processes are represented by distributions over strings of symbols representing business activities, and process mining algorithms as learning these distributions. This provides a unified probabilistic framework for considering the learning behaviour of different algorithms, regardless of their biases, assumptions, and the process representations which they use. This framework provides the formal basis within which to investigate the questions introduced in the previous sections, in particular to objectively compare the behaviour of mining algorithms and process models. It allows the completeness of an event log to be understood, and therefore quantification of the confidence that can be placed in process mining results.

In Chapters 5 and 6 we show the theoretical value of our framework, by using it to analyse two fundamental process mining algorithms. We show that probabilistic rules can be derived for the behaviour of each algorithm, in the form of formulae for the probability of correct mining of process 'sub-structures', from an event log of a given size, produced

by a known underlying process. We extend these results to mining full process models, and verify experimentally using two representative artificial process models.

In the next two chapters we turn to two practical process mining problems. Firstly, in Chapter 7 we consider process mining when the underlying process is changing, and show that our framework enables a method for detecting changes which are significant, in either the structure of or probabilities in the underlying business process, and recovery of the sequence of changed models. Secondly, in Chapter 8 we investigate mining from 'noisy' process data. We propose a formal model of 'noise' in process mining, missing from the existing literature, and analyse its effect on mining with the Heuristics Miner algorithm [194]. We show that our framework provides a basis for understanding the effect of 'noise' on the algorithm's behaviour, and therefore of how much data should be used for mining, and the effect of the algorithm's parameters.

Finally in Chapter 9 we discuss the theory and applications presented in the foregoing chapters, and evaluate the extent to which the thesis meets its aims. We summarise the contributions of the work and assess its limitations. We conclude by outlining some further questions raised by this work and suggest directions for future research.

1.6 Scope of this Thesis

In this thesis we consider only processes without cycles (loops), thus ensuring that all distributions have finite support. We assume that processes have single input (start) and output (end) activities, and the events of activities' occurrence are recorded as they occur. Events are atomic (take no time), and are uniquely labelled, the same label always referring to the same event, and vice versa. No use is made of additional information (such as timing) about events, merely the order in which they are recorded. The underlying process model is assumed to be fixed[1] (unlikely in reality, but change is assumed to be slow enough to be ignored over the period that data is collected). These restrictions

[1]This basic assumption is retained in Chapter 7 when investigating non-stationary processes.

are equivalent to those used elsewhere in the literature, e.g. [7, 39, 156]. We also do not set out to create new or improved mining algorithms, although the analyses of existing algorithms may provide insights into possible improvements. Future work could remove these restrictions without fundamentally changing the presented approach.

1.6.1 Success Criteria

The success criteria against which we evaluate this framework, are

1. coherent analyses of current, relevant process mining algorithms which explain and provide insight into their behaviour, and
2. successful solutions to practical process mining problems, under this framework, through application of the presented theory.

1.7 Contributions of this Thesis

Here we include a brief index to the main results in this thesis.

1.8 Publications Resulting from this Thesis

Parts of the research in this thesis have been published in several papers:

[189] P. Weber, B. Bordbar and P. Tiňo.
A principled approach to mining from noisy logs using Heuristics Miner. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 119-26, 2013.

[**23**] B. Bordbar and P. Weber.
Automated prevention of failure in complex and large systems: Fighting fire with fire. Accepted for publication in *International Journal of Informatics Society (IJIS)*, 5:(to appear), 2013.

[**184**] P. Weber, B. Bordbar and P. Tiňo.
A framework for the analysis of process mining algorithms. *Systems, Man and Cybernetics: Systems, IEEE Transactions on*, 43(2):203–317, 2013.

[**190**] P. Weber, P. N. Taylor, B. Majeed and B. Bordbar.
Comparing complex business process models. In *2012 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2012.

[**188**] P. Weber, P. Tiňo and B. Bordbar.
Process mining in non-stationary environments. In *20th European Symposium on Artificial Neural Networks (ESANN)*, 2012.

[**185**] P. Weber, B. Bordbar and P. Tiňo.
A principled approach to the analysis of process mining algorithms. In H. Yin, W. Wang, and V. J. Rayward-Smith, editors, *IDEAL*, volume 6936 of *Lecture Notes in Computer Science*, pages 474–481. Springer, 2011.

[**186**] P. Weber, B. Bordbar and P. Tiňo.
Real-time detection of process change using process mining. In A. V. Jones, editor, *ICCSW*, volume DTR11-9 of *Department of Computing Technical Report*, pages 108–114. Imperial College London, 2011.

CHAPTER 2

BUSINESS PROCESS MODELLING

In this chapter we discuss business process modelling from a process mining perspective. We focus on the main representations used by process mining algorithms to represent business processes, relating them to our running example process.

Business processes describe the activities carried out to fulfil a business function, and the relations between them [76]. Such functions include providing a service to customers [68] or producing a product [126], making purchases and handling invoices [152], financial management [84] or dealing with customer complaints.

Traditionally, business processes have been viewed as languages over activities, with no probabilistic structure. Various representational mechanisms have been suggested for capturing the control flow of business processes. These range from formal languages such as Petri nets [107, 112, 157] or BPMN diagrams [109] which allow systematic analysis and comparison, to flowchart notations used to informally discuss business processes.

We begin this chapter by reviewing Petri nets, the most common representation used in process mining research. We also introduce Causal Matrices and Heuristics Nets, used by the Heuristics Miner algorithm [194] (Chapter 6). We then introduce probabilistic automata (PDFA) and their relation to Petri nets. PDFA are the main representational framework which we use in this thesis to discuss business processes and process mining algorithms. We close the chapter by reviewing the main other process representations.

2.1 Petri Nets

Petri nets [112], especially a subset known as Workflow (WF) Nets [157], are the most common process representation used by process mining algorithms (e.g. [20, 156, 195]). Petri nets allow concurrency to be explicitly modelled in a succinct way. Concurrency is a critical feature of business processes, distinguishing them for example from finite grammars, since sequences of activities in different parts of a process may be executed in parallel by different people, perhaps synchronising at particular points in the process. Petri nets are executable, with formal semantics, enabling formal analysis of processes. There are various types of Petri net, details of which including their properties and executable behaviour can be found in [112]. For a discussion of Workflow Nets, a restriction of Petri nets commonly used in business processes, see [156, 157].

In general, a marked *Petri Net* is a 4-tuple $N = (S, T, W, M)$, where T and S are finite sets of *transitions* and *places* respectively, such that $T \cap S = \emptyset$. $W \subseteq (S \times T) \cup (T \times S)$ is a *flow relation* defining the directed arcs of the graph, connecting places and transitions. M is a multi-set over S called a *marking* $M : S \to \mathbb{N}$, describing the distribution of *tokens* over places, defining the state of the process. The initial marking is M_0. The workings of the Petri net are defined by the marking and the firing of transitions. Transition t may *fire* when there is a token in each of its input places, whereupon a token is removed from each of the input places of t and a token added to each of the output places of t.

Figure 2.1 shows a Petri net $N_0 = (S, T, W, M_0)$, where $S = \{p_0, p_1, \ldots, p_9\}$, $T = \{i, a, b, \ldots, o\}$, $W = \{(p_0, i), (i, p_1), (p_1, a), \ldots, (o, p_9)\}$, $M_0 = (1, 0, \ldots, 0)$. Solid rectangles represent transitions, modelling process activities. Places are shown by circles, and tokens by black dots in places. As depicted in Figure 2.1, only transition i can fire, consuming the token in p_0 and creating one in p_1, thus enabling transition a.

Following the execution of a single transition, more than one following transition may be enabled to fire next. For example, in N_0, when place p_2 contains a token, either b or c is enabled to fire next. If b fires, a token is created in both places p_3 and p_4, enabling both d and e. Therefore either d may fire next, followed by e, or e first followed by d. In

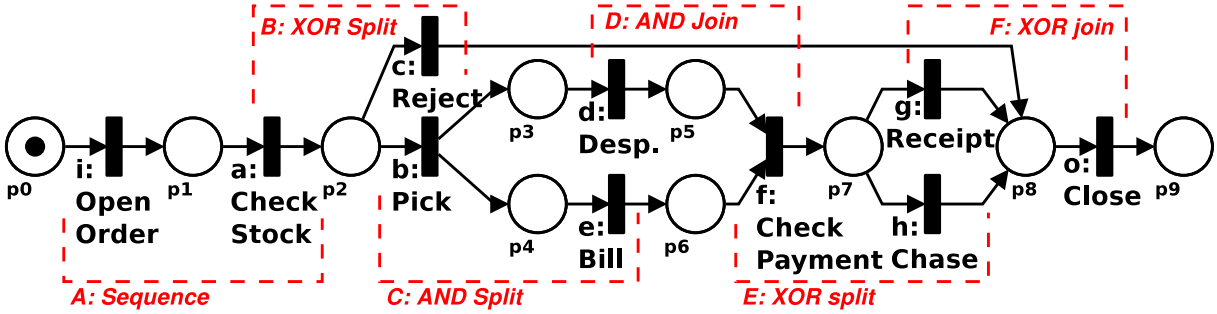


Figure 2.1: Petri Net N_0 for the Running Example, a simplified Business Process for fulfilling an Order, highlighting the Sub-Structures in the Process.

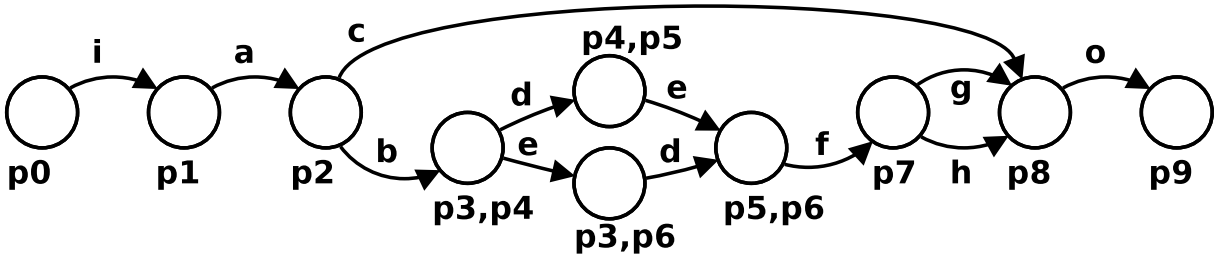


Figure 2.2: Reachability Graph for Petri Net N_0 (Figure 2.1).

this way the net models d and e as concurrent activities.

The *Reachability Set* $\mathcal{R}(N_0)$ of Petri net N_0 is the state space of the net, the set of markings reachable from M_0 by firing a series of transitions. Marking M' is *immediately* reachable from M if it can be reached by firing a single transition. A *reachable* marking M'' is a marking that is immediately reachable from M, or from marking M', itself immediately reachable from M. Figure 2.2 shows the *Reachability Graph* of Petri net N_0 (Figure 2.1), representing $\mathcal{R}(N_0)$ as a transition system. Each marking is represented by a state, labelled with the places of the Petri net which contain tokens in that marking. Arcs are labelled by the transitions fired to move from one state to the next.

2.1.1 Running Example as a Petri Net

Petri net N_0 (Figure 2.1) formally represents the simple running example process (Chapter 1, Figure 1.1). When an order is received, it is first registered in the order processing systems (i:open order). Next, stock is checked (a), then either the items picked from the warehouse (b), or the order rejected (c), shown by XOR split B. Despatch (d) and billing

(e) take place in parallel (AND split C), then after checking payment (f), either a receipt is issued (g) or payment chased (h), XOR split E. Finally the order is closed (o).

Clearly this is a simplified version of this process, which we use in this thesis to benchmark our analyses of process mining algorithms. In a real process we might expect that chasing payment (h) would be on a loop between f and g, allowing it to be repeated until payment is received. Alternatively there might be a separate sub-process for recovery of payment. In this thesis, however, we do not deal with cycles or hierarchical processes.

2.1.2 Workflow Nets

Sometimes business processes are constrained to a convenient subset of Petri Nets, called Sound Workflow (WF) Nets [124,156]. A *Sound WF-Net* is a Petri net with a single start and single end place and every transition on a path between these two places. Its marking is a mapping $S \rightarrow \{0, 1\}$, i.e. any place may hold at most one token. The initial marking M_0 is a single token in the start place, and final marking M_F a single token in the end place. When a process is started from M_0, all transitions must be potentially executable, and the process must terminate properly, i.e. in marking M_F. Sound WF-nets allow for all the basic routing constructs found in business processes.

Structured WF-Nets [156] restrict the allowed structure of places and transitions to ensure each split corresponds with a join of the same type. Figure 2.1 is both a Sound and a Structured WF-Net, in its initial marking M_0.

2.2 Causal Matrices and Heuristics Nets

Causal Matrices were introduced as a process representation for the Genetic Process Miner [47, 50] and Heuristics Miner [194] process mining algorithms. They were defined to be as expressive as Petri nets in showing dependencies between activities and the characteristics of splits and joins, without introducing constructs such as hidden transitions and places, needed by Petri nets to describe process sub-structures. Such constructs can

Activity	Input	Output
i	$\{\}$	$\{\{a\}\}$
a	$\{\{i\}\}$	$\{\{b, c\}\}$
b	$\{\{a\}\}$	$\{\{d\}, \{e\}\}$
c	$\{\{a\}\}$	$\{\{o\}\}$
d	$\{\{b\}\}$	$\{\{f\}\}$
e	$\{\{b\}\}$	$\{\{f\}\}$
f	$\{\{d\}, \{e\}\}$	$\{\{g, h\}\}$
g	$\{\{f\}\}$	$\{\{o\}\}$
h	$\{\{f\}\}$	$\{\{o\}\}$
o	$\{\{g, h, c\}\}$	$\{\}$

Table 2.1: Causal Matrix corresponding to the Running Example Process (Petri Net N_0, Figure 2.1), represented in the ProM Framework [170] as Heuristics Net (Figure 2.3).

be problematic for process mining algorithms, since there is no direct evidence for them in a Workflow log (Chapter 3).

A *Causal Matrix* is a 4-tuple $CM = (A, C, I, O)$, where A is a finite set of activities and $C \subseteq A \times A$ defines the relations between activities. $I : A \to \mathcal{P}\big(\mathcal{P}(A)\big)$ and $O : A \to \mathcal{P}\big(\mathcal{P}(A)\big)$ are functions describing the input and output conditions for each activity[1]. The I and O relations define the characteristics of splits and joins. For a given activity, I and O return a set of sets of activities (Table 2.1). Activities in a subset are exclusive (cannot occur together in a trace), whereas those in different subsets are in parallel. Table 2.1 shows the Causal Matrix for the running example, the process in Figure 2.1. The input to activity b is a single activity, a, while its output is d and e in parallel, thus defining a parallel (AND) split from b to d and e. Similarly, the input to f is d and e in parallel, while its output is g or h, defining a parallel join followed by an exclusive split. The Causal Matrix therefore defines the same process sub-structures as Petri net N_0. Indeed, a Petri net, with certain restrictions, can be mapped to a Causal Matrix, and *vice versa* [47, 50].

The ProM Process Mining Framework [170] depicts Causal Matrices graphically as Heuristics Nets, as shown in Figure 2.3 for our example process. Each node represents an activity[2], and nodes and arcs are labelled with the number of times they were involved in

[1] $\mathcal{P}(A)$ is the power set of A, the set of all subsets of A.

[2] ProM allows for 'begin' and 'complete' events associated with an activity, defaulting to 'complete' where only one is present, shown in the labelling of the nodes in Figure 2.3.

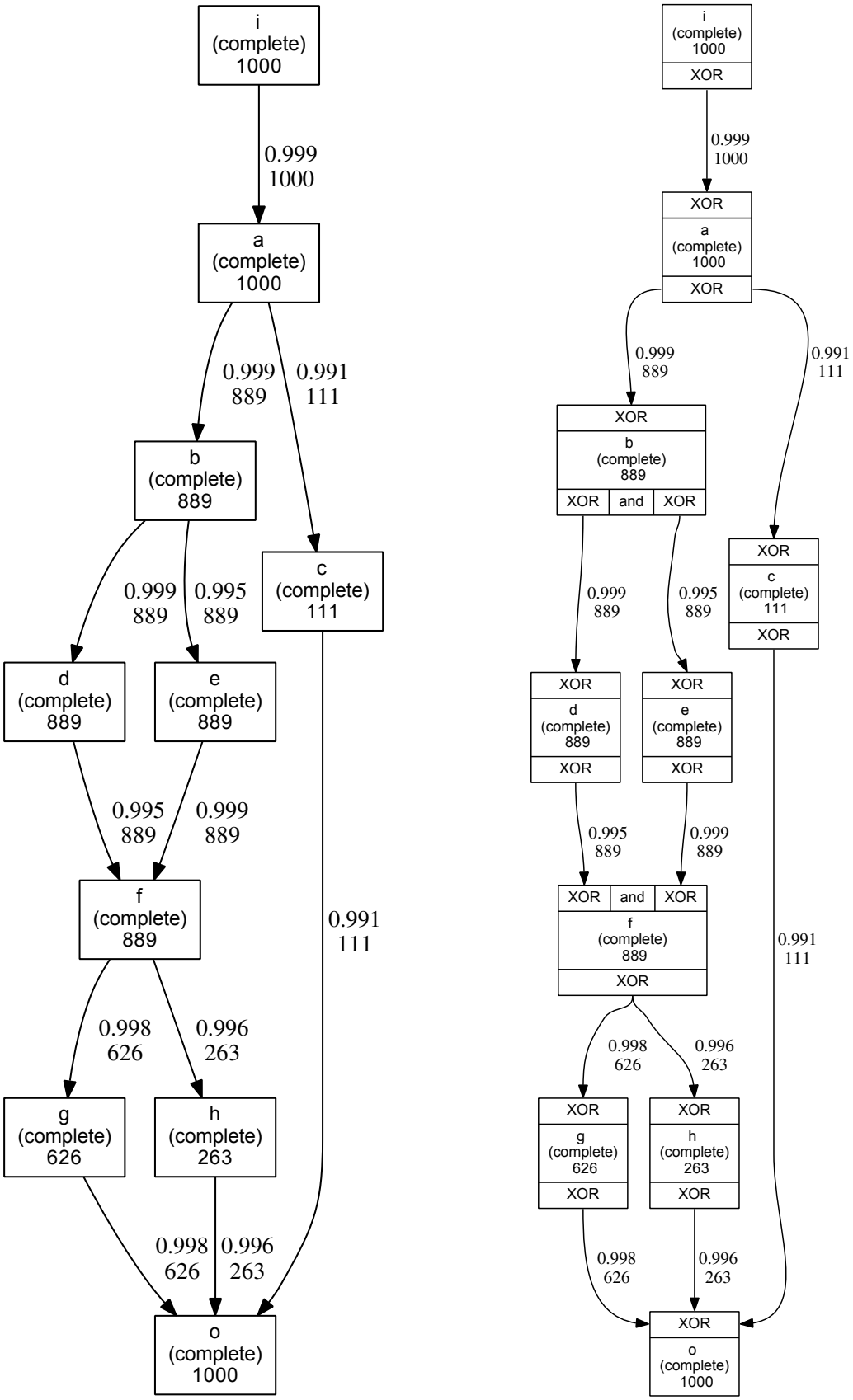


Figure 2.3: Heuristics Nets for the Running Example Process, as produced by the ProM Framework [170], with (right) and without (left) identifying the Types of Splits and Joins.

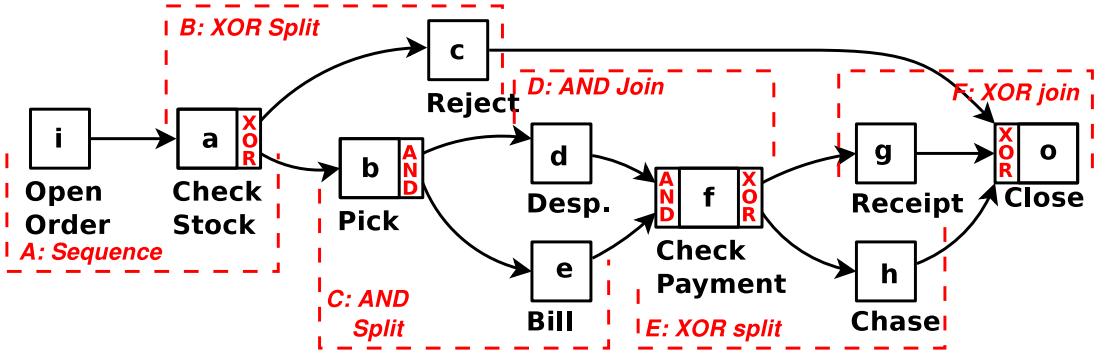


Figure 2.4: Running Example in the Heuristics Net Representation used in this Thesis.

parsing the log from which the model was mined. Optionally, splits and joins are labelled with their type, XOR (exclusive choice) or AND (parallel). In this thesis we use a compact representation showing the same information (Figure 2.4). Causal Nets [148, 193] have been recently proposed as a representation specifically designed for process mining. Causal Nets extend Causal Matrices, to merge the benefits of the formal semantics of Petri nets with the understandability and graphical representation of Heuristics Nets.

2.3 Probabilistic Automata

In this section we introduce Probabilistic Automata and relate them to Petri nets. Although representational limitations of such automata, especially their inability to explicitly represent concurrency, mean they are often not the most appropriate graphical representation for business processes (see e.g. [148]), they are the main representational framework which we will use in this thesis to discuss probability distributions generated by processes (Chapter 4).

A *PDFA* is a five-tuple $A = (Q_A, \Sigma, \delta_A, q_0, q_F)$:

- Q_A is finite set of states;
- Σ is an alphabet of symbols;
- $\delta_A : Q_A \times \Sigma \times Q_A \rightarrow [0, 1]$ is a mapping defining the conditional transition probability function between states, $\delta_A(q_1, a, q_2) = Pr(q_2, a|q_1)$, i.e. the probability to parse

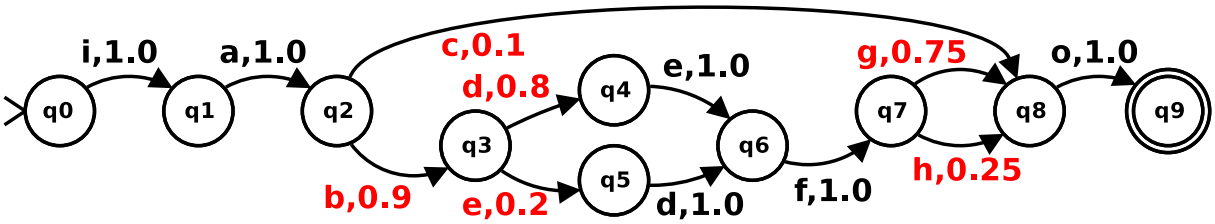


Figure 2.5: PDFA A_0 corresponding to Petri Net N_0 (Figure 2.1), with the addition of Transition Probabilities.

symbol a and arrive in state q_2 given currently in q_1;

- $q_0 \in Q_A$ is a single start state; and
- $q_F \in Q_A$ is a single end state; such that:

$$\forall q \in Q_A, \sum_{q' \in Q_A, a \in \Sigma} \delta_A(q, a, q') = 1,$$

and

$$Pr(q'|a, q) = 1.$$

The probabilities on arcs outgoing from a state sum to 1, and given a current state and symbol, the next state is certain. There is a unique state path $Q^{(x)}$ through the automaton for any string x that it can parse.

Example PDFA A_0 (Figure 2.5) represents the same model as Petri net N_0 (Figure 2.1). It has the same structure as the reachability graph (Figure 2.2), with the addition of probabilities of following each arc, or parsing each symbol. Here $Q = \{q_0, q_1, \ldots, q_9\}$, $\Sigma = \{i, a, b, \ldots, o\}$, $\delta = \{(q_0, i, q_1) \to 1.0, (q_1, a, q_2) \to 1.0, \ldots, (q_8, o, q_9) \to 1.0\}$, $q_0 =$‘q_0’, $q_F =$‘q_9’. States are shown by circles, the start state is indicated by an arrow and the final state by a double border. The state path to parse $x = iaco$ is $Q^{(x)} = \{q_0, q_1, q_2, q_8, q_9\}$.

Every PDFA A describes a distribution P_A over Σ^+:

$$P_A(x) = \delta_A(q_0, s_0, q_{s_0}) \times \left(\prod_{i=1}^{n-2} \delta_A(q_{s_{i-1}}, s_i, q_{s_i}) \right) \times \delta_A(q_{s_{n-2}}, s_{n-1}, q_F),$$

where x is a string of symbols $s_0 s_1 \ldots s_{n-1}$ which can be parsed by the automaton to the unique final state q_F; q_{s_i} denotes the state reached after symbol s_i is parsed. $P_A(x) = 0$

for strings which cannot be parsed.

In A_0 (Figure 2.5), $P_A(iaco) = \delta_A(q_0, i, q_1) \times \delta_A(q_1, a, q_2) \times \delta_A(q_2, c, q_8) \times \delta_A(q_8, o, q_9) = 1.0 \times 1.0 \times 0.1 \times 1.0 = 0.1$.

Note that the structure of allowed traces defined by sound WF-Nets can be naturally captured by the support structure of distributions described by PDFA in the sense that

- there is a single start and end state,
- all states are accessible (reachable from the initial state),
- from any state, it is possible to reach the final state,
- for any given string x, the sequence of state transitions to generate x is unique, and
- given a state and a symbol, the next state is certain.

PDFA impose only a weak representational bias on a process model, whereas processes tend to be structured. PDFA also cannot succinctly represent concurrency, an important characteristic of business processes and problem for process mining algorithms. For example, in Figure 2.5, PDFA A_0 explicitly represents parallel activities d and e as alternative paths $d \rightarrow e$ and $e \rightarrow d$. As the number of parallel activities increases, the number of alternate paths to explicitly represent grows exponentially, cluttering the graphical representation (see for example Figure 5.17, Chapter 5).

PDFA are therefore not favoured (e.g. [148]) for representing real business processes, which may be large and complex, with many activities and relations between them, structured, and involve concurrency. However, PDFA are a useful representation for the analysis which we carry out in this thesis. A sound WF-Net does not hold any probability information, but has finite state space, so the net's structure can be converted to an automaton with a finite number of states, as discussed in the following sub-sections.

2.3.1 Conversion of Petri Nets to Probabilistic Automata

A Sound WF-Net N holds no probability information, but its structure can be converted to a PDFA via its Reachability Set $\mathcal{R}(N)$, and arc probabilities estimated, e.g. from

sample data. The 'state space explosion' may be a problem, especially in the conversion of large Petri Nets with high concurrency.

Let N be a marked Sound WF-Net $N = (S, T, W, M_0)$, with single start and end transitions $t_i, t_o \in T$, initial and final markings M_0, M_F, and reachability set $\mathcal{R}(N)$. There exists a structurally equivalent probabilistic automaton $A = (Q_A, \Sigma, \delta_A, q_0, q_F)$:

- $Q_A = \mathcal{R}(N)$,
- $\Sigma = T$,
- $\delta_A : q \times t \times q' \to 1/n(q)$, $q, q' \in \mathcal{R}(N)$, $t \in T$ is enabled in marking q and firing t moves the marking to q'. $n(q)$ is the number of arcs leaving q in the PDFA structure.
- $q_0 = M_0$,
- $q_F = M_F$.

This definition of δ_A labels all transitions leaving a state, with equal probability, thus assigning uniform probability to the paths following any exclusive or parallel split. Alternatively, maximum likelihood probabilities can be estimated from data.

2.3.2 Conversion of Probabilistic Automata to Petri Nets

The structure of a probabilistic automaton may be converted to a Petri Net using the Theory of Regions (see e.g. [164]). Given PDFA $A = (Q_A, \Sigma, \delta_A, q_0, q_F)$, a *region* is a subset of states $Q' \subseteq Q_A$ such that for each $a \in \Sigma$, one of the following holds true:

- all transitions $\delta(q_1, a, q_2) > 0$ enter Q' from outside, i.e. $q_1 \notin Q' \wedge q_2 \in Q'$, or
- all $\delta(q_1, a, q_2) > 0$ exit Q' from inside, i.e. $q_1 \in Q' \wedge q_2 \notin Q'$, or
- all $\delta(q_1, a, q_2) > 0$ do not cross the boundary of Q', i.e. $q_1, q_2 \notin Q' \vee q_1, q_2 \in Q'$.

Region Q' is a *sub-region* of another region Q'' if $Q' \subset Q''$, and a region is *minimal* if it contains no subregions. Q' is a *pre-region* of symbol $a \in \Sigma$ if $\exists\, q_1 \in Q', q_2 \notin Q' : \delta(q_1, a, q_2) > 0$. Q' is a *post-region* of a if $\exists\, q_1 \notin Q', q_2 \in Q' : \delta(q_1, a, q_2) > 0$.

There exists a marked Petri Net $N = (S, T, W, M_0)$ structurally equivalent to PDFA A, with set of minimal regions R:

- $S = R$,
- $T = \Sigma$,
- $W = \{(s,t) \in (S \times T) \cup (t', s') \in (T \times S) : s \in R$ is a pre-region of t, $s' \in R$ is a post-region of t', and
- $M_0 = (1, 0, 0, \ldots, 0)$.

N may not be minimal. The Petri Net synthesis literature discusses what transition systems may be converted to Petri Nets and how they can be reduced. For discussion and references in the context of process mining, see [164].

2.4 Other Business Process Representations

While Petri nets, as introduced in Section 2.1, are the most common representation used in the research literature relating to process mining and the analysis of business processes, various representations have been used to capture the control-flow of processes (see e.g. [148, Section 2.2]). Transition systems, used by some early approaches to process mining, [39, 44], simply represent states of the process and the activities which cause the process to change state. They allow reasoning about the process behaviour, but are unable to express concurrent behaviour succinctly since every state must be represented (Section 2.3). YAWL ('Yet Another Workflow language') [166] was developed to allow many more workflow patterns [154, 196] than those easily representable by Petri nets, such as non-exclusive OR splits and joins, but has not been used for process mining.

While these representations allow models which are formally analysable, some constructs such as Petri net places cannot be directly derived from event (workflow) logs, and must be inferred. Many process mining algorithms (e.g. [73, 194]) therefore use simple directed graph notations in which nodes represent activities and edges the causal dependencies between them. These are intuitive to understand, but may not be rigorously analysable. Causal Nets [148, 193] have recently been introduced as a representation specifically designed for process mining, merging the benefits of formal semantics

with understandability. Only a very few, non-mainstream, algorithms use probabilistic representations, e.g. Hidden Markov Models [80], probabilistic automata [63].

Industry often favour easily-understandable representations. Event-driven Process Chains (EPCs) [87] are used by the Multi-Phase Miner in relation to the SAP Enterprise Resource Planning (ERP) software [54, 171]. Notably they allow non-exclusive OR splits, but their semantics are problematic due to unclear specification [161]. Business Process Modelling Notation (BPMN) [109], standardised by the Object Management Group (OMG) is currently favoured by practitioners and vendors. It provides for hierarchical models and a rich and flexible notation. Business Process Execution Language (BPEL) [88] is a non-graphical language for describing processes, particularly in the context of Web Services. The Unified Modelling Language (UML) [129] state, activity and sequence diagrams are used for object-oriented modelling, including of business processes.

Mappings exist between subsets of many of these representations [95]. For example, BPEL to Petri nets [81, 139] and Workflow nets [162], BPMN to Petri nets [116], Petri net to EPC [177], and UML diagrams to Petri nets [17].

In Table 2.2 we summarise the main process representations used in process mining, by the process mining algorithms which make use of them.

2.5 Chapter Summary

In this chapter we reviewed the main representations used to model business processes, concentrating on Petri nets, Causal Matrices and Heuristics Nets. We introduced Probabilistic Automata as our main representational framework for discussing business processes and process mining algorithms. This discussion supports the next chapter, in which we introduce process mining and the problem addressed by this thesis, that of comparing and analysing process models and mining algorithms. The multitude of process representations contributes to this problem.

Representation Language	Algorithms/Authors
Automata	
Process Activity Graph	Datta [44]
Finite State Machine	Datta [44], RNet, KTail, Markov [39], Two-Step Approach [164]
Probabilistic	
Probabilistic Automaton	Expectation-Maximisation Algorithm (Unlabelled Traces) [63]
Hidden Markov Model	Herbst [80]
Other Directed Graphs	
Simple Directed Graph	Agrawal [7], Genetic Programming [145]
Causal Matrix	Heuristics Miner [194], Heuristics Miner^{++} [28, 29], Genetic Miner, [46, 47, 49, 50, 151], Simulated Annealing [138]
Heuristics Net	Heuristics Miner [194], Heuristics Miner^{++} [28, 29], Genetic Miner [46, 47, 49, 50, 151], Context-Aware Trace Clustering [24, 25]
Causal Net	Flexible Heuristics Miner [193]
Adonis Definition Language	InWoLvE [78–80]
Block-Structured Model	Schimm [133]
Process Languages	
Event-Driven Process Chain (EPC)	Multi-Phase Miner [54, 171]
Workflow Model Graph	Interval Sorted Algorithm [113]
Statistical Dependency Table	WorkFlowMiner [65]
Types of Petri Net	
Structured Workflow (SWF) Net	Alpha Algorithm [156], Alpha^{+} [48], Alpha^{++} [195]
Other Petri Net	Alpha#, γ ,τ, γ^{+} [199], Beta [117], Region Miner [20, 55], Two-Step [164], Agnes [67], Trace Clustering [137]
S-Coverable Workflow Net	Algorithm S [85]
History-Dependent Stochastic Petri Net (HDSPN)	Predictive analysis [134]
Informal Representations	
Fuzzy model/Process Map	Fuzzy Miner [73, 74, 163], Clustering [27, 91]
Simple Precedence Diagram	Visualisation and Clustering [172]
Other Representations	
Declarative Language	Declare [97]
Markov Logic Network	DPML, Alchemy [19, 97]
Workflow Schema	Hierarchy Discovery [51, 70, 71]
Self-Organising Map	Trace Clustering [137]

Table 2.2: Process Representations by Process Mining Algorithm.

CHAPTER 3

BUSINESS PROCESS MINING

Having introduced business process modelling and process representations, we turn in this chapter to business process mining, and introduce the problems addressed by this thesis.

We introduce process mining and the standard view of process mining algorithms. We then discuss two problems with this view: non-determinism in business processes, and the many different process mining algorithms, representations and comparison metrics. The first problem leads to uncertainty in the correctness of models output by process mining algorithms; the second makes it difficult to compare these models and algorithms.

Process mining is essentially a machine learning task, but little work has been done on systematically analysing mining algorithms in this context, to discover their fundamental properties, or to answer questions such as how much data is necessary for mining. Yet such understanding is of critical importance to give confidence that an event log is an adequate sample of the true behaviour, and thus in the correctness of the mined model.

In the remainder of the chapter we describe two common mining algorithms, the Alpha Algorithm [156] and Heuristics Miner [194], which are analysed under our framework in later chapters. To motivate the framework proposed in this thesis, for the analysis and comparison of process mining algorithms, we review the main other process mining algorithms, and metrics used for comparing process models.

Start Time	Case ID	User ID	Activity	Other Data
2013-07-01 12:00	0001	AB	*i*:Open Order	orderno
2013-07-01 15:30	0002	CD	*i*:Open Order	orderno
2013-07-01 15:35	0001	AB	*a*:Check Stock	
2013-07-02 9:20	0002	AB	*a*:Check Stock	
2013-07-02 11:00	0001	GH	*b*:Pick	stock level
2013-07-02 11:30	0001	GH	*d*:Despatch	
2013-07-03 9:00	0001	Fin1	*d*:Issue Bill	terms
2013-07-03 10:15	0002	GH	*c*:Reject	
2013-07-03 17:05	0003	AB	*i*:Open Order	orderno

Table 3.1: Example of Event Logs from the Running Example Process.

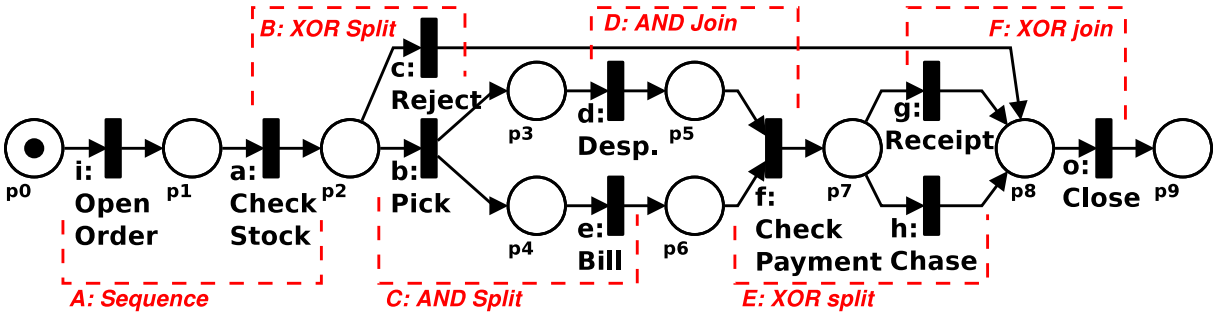


Figure 3.1: Simplified Business Process for fulfilling an Order.

3.1 Business Process Mining

Businesses may design business processes, introduced in Chapter 2, to dictate work patterns. Alternatively, business processes may result *de facto* from a company's working practices. Either way, as activities take place, the information systems involved record information in 'workflow' or 'event' logs. Process mining uses these logs to discover and analyse models of business processes.

In Table 3.1 we show an example of information which might be recorded for the running example process (shown as a Petri net in Figure 3.1). Each record contains information about a process event corresponding to one of the activities in the model, 'Open Order', 'Check Stock', etc. For each event is recorded: Start Time, Case ID identifying activities belonging to the same run through the process, a User ID, and perhaps additional information. Attributes not shown might include activity end time or department ID. Table 3.1 shows the first few activities from three process cases running

Process Trace
$iabdefgo$
$iabdefho$
$iabdefho$
$iabdefgo$
$iabdefgo$
$iabdefho$
$iaco$
$iabdefgo$
$iabedfgo$
$iabdefgo$
$iabdefgo$
$iabdefho$

Table 3.2: Part of Event Log of Traces randomly simulated from the Running Example Process, represented as Strings.

at the same time. This information might be stored in a database, in a homogeneous set of logs in a standard format (e.g. from an ERP or CRM system), or in heterogeneous text files produced by unrelated information systems. Process mining algorithms assume that a pre-processing step has collated the information into a standard event log format such as MXML [167] or XES [178].

Processes may be described from various 'perspectives' [155], including relationships between activities (control-flow), timing, resources, or decision rules. In this thesis we focus on the control-flow perspective. Abstracting from detail in the log, the 'traces' of enactments of the process might be recorded in an event log as strings such as *'iabdefgo'*, *'iaco'*, where each symbol represents one of the activities in the model. An event log is thus represented by a multiset of such strings (Table 3.2)

Process discovery algorithms use logs of such traces to produce models of process control flow. Process mining also addresses performance analysis [172], troubleshooting, auditing conformance [12, 122, 124], mining decision rules [128], or interaction between resources [153]. A current focus is on managing complex processes or event logs, by abstracting from detail or separating multiple processes recorded together [24, 70, 73, 74, 137,172]. More detail on process mining may be found in [148] and the reviews in [144,155].

We next introduce our notation for discussing business processes and process mining,

then outline the standard view of Process Discovery, and two problems with the standard view which we address in this thesis. In the remainder of the chapter we describe in detail two common process mining algorithms, and review other mining algorithms and process mining metrics used for comparison of process models.

3.1.1 Notation

In this section we introduce notation used in this thesis to discuss business processes and process mining (summarised in Table 3.3). The notation will be formally described throughout the remainder of this chapter and the next.

We consider a set of business activities $\mathcal{A}$ and an underlying business process $\mathcal{M}$ defined over $\mathcal{A}$. In this thesis we propose a probabilistic view of process mining (Chapter 4). We represent a process by a probability distribution P_M over non-empty strings $\{x, y, \ldots\} \in \Sigma^+$ of symbols $\{a, b, \ldots\} \in \Sigma$ from a finite alphabet representing activities in $\mathcal{A}$. Such strings represent process traces. We write $\pi(w)$ as shorthand for the probability of sub-string w occurring in a process trace.

$\mathcal{T} \subset \Sigma^+$ is a subset of such strings, representing valid process traces. The set $\mathcal{T}$ is finite, since we consider only acyclic models and impose that traces begin with symbol i and end with symbol o. A Workflow or Event log $\mathcal{W}$ is therefore a multiset of strings $x \in \mathcal{T}$. We write xy to describe concatenation of strings x and y, $x\Sigma^*$, $\Sigma^* x$ and $\Sigma^* x \Sigma^*$ for the sets of strings with x as prefix, x as suffix and x as sub-string, respectively.

We also consider frequencies $N(x)$ with which strings x occur in a log $\mathcal{W}_n$ of n traces. We derive in this thesis formulae for the probability of correctly mining process sub-structures S (e.g. sequences of activities, splits and joins) from $\mathcal{W}_n$. These probabilities are denoted $P_{\alpha,n}(S)$ for the Alpha Algorithm [156] (Section 3.2.1), $P_{HM,n}$ for the Heuristics Miner algorithm [194] (Section 3.2.2). Table 3.3 includes algorithm-specific notation for process sub-structures and probabilities, which we will introduce in Chapters 5 and 6.

$\mathcal{A}$	A set of business activities.
$\mathcal{M}$	'Ground truth' model (may be unknown).
Σ	Alphabet of symbols encoding business activities.
$\{a, b, \ldots\} \in \Sigma$	Valid business activities in the process.
$\{x, y, \ldots\} \in \Sigma^+$	Non-empty strings representing sequences of activities.
$\mathcal{T}$	The set of all valid process traces (cases).
xy	The concatenation of strings x and y.
$x\Sigma^*, \Sigma^* x,$	The set of strings with x as prefix, suffix,
$\Sigma^* x \Sigma^*$	... or sub-string. (rest of string may be empty).
$\mathcal{W} \subset \{x \vert x \in \mathcal{T}\}$	Event (Workflow) log, a *bag* or *multi-set* of traces.
$\mathcal{W}_n$	Event log containing n traces.
$N(x)$	Number of times x occurs in $\mathcal{W}$.
P_M	Probability distribution over traces, describing process $\mathcal{M}$.
$\pi(w)$	Probability of sub-string w occurring in a trace.
$\pi(\rightarrow a)$	Probability of 'reaching' a in the model: $\pi(\rightarrow a) = \pi(i\Sigma^* a \Sigma^* o)$.
$a \rightarrow b$	Arc representing causal dependency from a to b.
$a \rightarrow (b_1 \parallel b_2 \ldots)$	Parallel ('AND') split, from a to paths starting with $b_1, b_2, \ldots$.
$a \rightarrow (b_1 \# b_2 \ldots)$	'XOR' split from a to alternative paths starting with $b_1, b_2, \ldots$.
$\mathrm{DM}_{ab} \in [-1, 1]$	Heuristics Miner (HM) Dependency Measure (DM) between a and b.
$\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$	Probability that HM constraint for correctly mining a structure holds true from event log of n traces; also e.g. $\gamma_n\big(N(ia) > \mathrm{PO}\big)$.
$P_{\alpha,n}(S)$	Probability that Alpha mines structure S correctly from $\mathcal{W}$.
$P_{HM,n}(S)$	Probability that HM mines structure S correctly from $\mathcal{W}$.

Table 3.3: Notation for Business Processes and Process Mining.

3.1.2 Standard View of Process Discovery

In Figure 3.2 we illustrate the standard view of process discovery. An underlying business process model $\mathcal{M}$ is assumed, which controls what activities take place. There may also exist an 'assumed' or ideal business process model $\mathcal{M}^*$ which describes how the business requires or believes the process to operate, perhaps resulting from business analysis or process design activities. Alternatively, no such process may be known.

As business activities take place, information is recorded in an event log $\mathcal{W}$. Process mining algorithms use the 'evidence' in $\mathcal{W}$ to construct a mined model $\mathcal{M}'$, using some notation such as a Petri net (e.g. Figure 3.1). The recovered model $\mathcal{M}'$ is assumed to accurately represent the underlying process $\mathcal{M}$. The recovered and 'believed' models $\mathcal{M}'$ and $\mathcal{M}^*$ can then be analysed to understand the reasons for divergence from the required

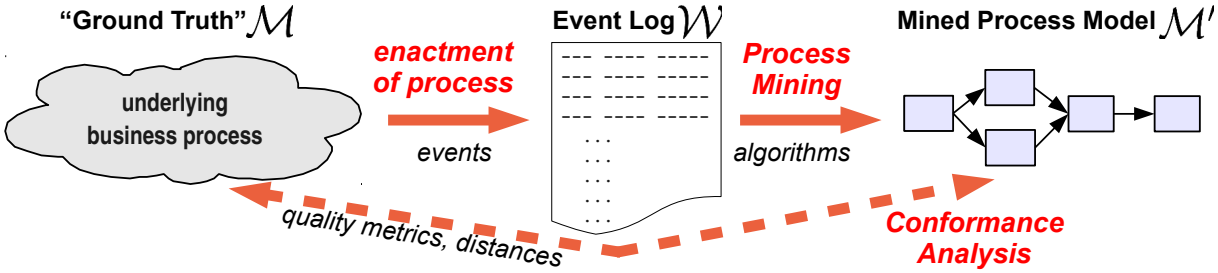


Figure 3.2: Standard View of Process Discovery.

Process Trace	**Trace Counts in 1000 Trace Log**									
$iabdefgo$	529	583	543	544	540	538	541	541	545	531
$iabdefho$	185	160	185	195	190	184	189	186	178	179
$iabedfgo$	140	110	129	123	119	146	118	135	119	150
$iabedfho$	44	40	39	37	50	37	54	39	46	43
$iaco$	102	107	104	101	101	95	98	99	112	97

Table 3.4: Counts of Traces (represented as Strings), in 10 Event Logs of 1000 Traces randomly simulated from the Running Example Process.

process, investigate performance or understand decision rules. When no $\mathcal{M}^*$ is known, $\mathcal{M}'$ provides a first understanding of the 'true' business process, and can be used as a basis for developing process improvements.

Since the core interest is in process control flow, many algorithms learn only the process structure, without attempting to recover probabilities. In the next sections we look at two problems with this standard view of process discovery.

3.1.3 Problem 1: Process are not Deterministic

Probabilities in the model may be of interest, for example where business rules restrict the frequency of costly patterns of activity. However even where there is no interest in producing a probabilistic model, it must be appreciated that traces are generated randomly according to an underlying probability distribution unknown to the mining algorithm. Not all activities or decisions are equally likely, and their probabilities may have a dramatic effect on the amount of data needed for mining.

As an example, Table 3.4 shows the counts observed of the five possible traces supported by the running example process, from 10 event logs of 1000 traces each, randomly

simulated from the model. The observed counts vary between event logs, and it is conceivable that an event log might contain no examples of a possible trace. This is particularly a problem with parallel activities: n parallel activities may be recorded in the event log in $n!$ possible orders, some of which may have very low probability.

Noise is also a source of non-determinism in event logs. Noise in process mining is not well defined, but includes problems in recording the event log (missing activities or parts of traces, or activities recorded out of order), which will affect the probability of 'correct' traces in the event log and introduce 'incorrect' traces. We discuss noise in Chapter 8.

Traces in $\mathcal{W}$ therefore represent only samples of the possible process behaviour. As the process executes, traces are generated randomly according to the underlying 'real-world' process $\mathcal{M}$, which describes the probability with which each sequence of activities (process trace) will appear in $\mathcal{W}$. It could be argued that traces do not appear randomly, but rather their frequency in $\mathcal{W}$ is the result of many deterministic factors, such as business rules, user preferences, market conditions, customer activity, season, and so on. However, the complex interaction of these factors is effectively impossible to analyse. As a useful abstraction we can state that a process trace t will occur with probability $P_M(t)$ given by a probability distribution P_M over traces, describing the process $\mathcal{M}$.

The goal for a process discovery algorithm $\mathcal{L}$ is to use the evidence for the underlying process $\mathcal{M}$, recorded in event log $\mathcal{W}$, to recover a model equivalent to the underlying process model $\mathcal{M}$. The recovered model $\mathcal{M}'$ will only be an approximation to $\mathcal{M}$, since it depends on the random sample $\mathcal{W}$, the inductive biases of $\mathcal{L}$, and the representational bias of the modelling notation used by $\mathcal{L}$, as follows

Representational bias refers to limitations imposed by the modelling notation [148, Sections 5.4.1, 6.1.1]. For example, the Alpha algorithm [156] (Chapter 5) assumes that the underlying process is representable by a restricted Petri net known as a SWF-Net; algorithms using finite automata cannot explicitly represent parallel behaviour; and many algorithms insist that all nodes are labelled, and the labels are unique [159].

Inductive Bias refers to the set of assertions or assumptions made by the mining algorithm to restrict the hypothesis space from which to select a process model [104, Section 2.7]. The Alpha algorithm for example assumes that the behaviour of the process can be completely described by pairs of activities seen in $\mathcal{W}$. Other common assumptions include that events are atomic (taking no time), are uniquely labelled (the same label always refers to the same event and vice versa), and make no use of additional information such as timing of events, merely the order in which they are recorded.

Therefore to ensure confidence that $\mathcal{M}'$ is a good approximation to $\mathcal{M}$, it is crucial to understand the nature of the distribution over traces $P_M(t)$, and how the behaviour and biases of the algorithm $\mathcal{L}$ determine how $\mathcal{L}$ uses the evidence in $\mathcal{W}$ to produce $\mathcal{M}'$.

3.1.4 Problem 2: Heterogeneous Mining Algorithms, Representations and Metrics

Process mining is a relatively young research area [155]. New algorithms continue to be developed with different biases or theoretical foundations, or aimed at mining in specific situations. This raises the question of which algorithm is best in a particular situation, and how to compare algorithms.

In Chapter 2 we saw that various representational mechanisms have been suggested for capturing process control flow. Traditionally, business processes have been viewed as languages over activities, with no probabilistic structure. Many process mining algorithms produce types of Petri net, but algorithms produce results in many different representations, introducing different biases in the models they produce. This heterogeneity of modelling representations means that it is difficult to compare results from different algorithms. Also, since most models are non-probabilistic, they cannot be used directly to understand the learning behaviour of algorithms. It is highlighted in [158] – 'the world is NOT a Petri net' – that whatever the representation, the model is always an abstraction and may not represent reality. Process mining literature however often reads

as if the purpose of a mining algorithm is to discover 'the' underlying Petri net. This is exemplified by discussions of the ability of an algorithm to discover representation-specific features which cannot be present in event logs, such as 'invisible' activities or duplicate labelling of nodes.

Various methods have been proposed for comparing process models, evaluating a process model against an event log from which it was mined, or comparing a mined model with a reference model. These metrics are often applicable only to a specific algorithm or based on the syntax of a particular process representation. Examples are replaying training or reference logs [46,70,73,194], measuring Petri net token behaviour [12,67] or string edit distances [40], comparing incidence matrices [16] or coding costs using the Minimum Description Length principle [32]. Measures may be along different 'dimensions' [122,124] depending on the type of differences to be measured.

Many of these metrics are specific to the syntax or semantics of a particular representation, are difficult to interpret, and do not allow for the comparison of models in different representations. In general they do not consider the significance of differences. The underlying process is stochastic (Chapter 4): sequences of activities (traces) occur with specific probabilities, so there will be random variation in the distribution of traces observed in different event logs produced by a process. This variation may cause differences between models mined from different event logs from the same process. The values calculated for measures of difference between models therefore depend on factors such as characteristics of the models, underlying event probabilities, how much data was used and the learning behaviour of the mining algorithm.

We therefore face the problem of how to compare heterogeneous algorithms which produce process models in a variety of incompatible representations. In the next section we look in detail at two process discovery algorithms, the Alpha Algorithm [156] and Heuristics Miner [194] which are analysed in later chapters under the framework presented in Chapter 4, then review the main other process mining algorithms and process metrics.

3.2 Common Process Mining Algorithms

In this section we describe the Alpha [156] and Heuristics Miner [194] Algorithms.

The Alpha algorithm was designed to handle concurrency in processes, and proven to correctly mine processes where the underlying process can be modelled by a Structured Workflow net (SWF-Net), a subclass of Petri net (Chapter 2). The algorithm is simple to apply, and although considered unsuitable for 'real-world' processes, is often used to obtain a first insight into a process. Alpha requires a 'complete' event log, where completeness means that for every pair of activities that the model allows to directly follow each other, there is a trace in the log that exhibits this behaviour. Alpha uses such pairs of activities identified from the event log, to attempt to exactly replicate the underlying process from the traces in the event log. Therefore it is unable to handle 'noise'. We discuss in Chapter 8 definitions of noise in process mining. For now, we note that process, system and data problems may cause Alpha to produce a difficult to understand, or 'spaghetti' [148] model.

Heuristics Miner is explicitly designed to handle 'noise' in event logs, and for this reason has been the algorithm of choice in many practical applications [35, 191], where processes have been found to be complex and poorly recorded. Examples are local government [152], healthcare [99], finance [45] and telecoms [67]. Whereas Alpha makes hard decisions based on the presence or absence of pairs of activities in the event log, Heuristics Miner uses counts of pairs of activities, and differences between counts of different pairs of activities, to determine which arcs to create in the process model. Several parameters to the algorithm give control over these softer decision boundaries, and thus of the detail to include in the mined model.

In Chapters 5 and 6 respectively we apply our framework to these two algorithms.

3.2.1 Alpha Miner Algorithm

Consider two events a and b from the set of activities $\mathcal{A}$ belonging to a process $\mathcal{M}$ recorded in an event log $\mathcal{W}$. Let $N(ab)$ denote the number of times sub-string ab occurs in traces

in $\mathcal{W}$. We say sub-string ab occurs in event log $\mathcal{W}$ ($ab \in \mathcal{W}$) if $N(ab) > 0$. If $ab \in \mathcal{W}$, then b directly follows a in at least one trace, which we write as $a > b$. We define the following four possible relations between a and b:

- $a \rightarrow b \iff N(ab) > 0$ and $N(ba) = 0$;
- $b \rightarrow a$ (which we can also write $a \rightarrow^{-1} b$);
- $a \# b \iff N(ab) = N(ba) = 0$;
- $a \parallel b \iff N(ab) > 0$ and $N(ba) > 0$.

Two activities are always related by either $\rightarrow$, $\rightarrow^{-1}$, $\#$ or $\parallel$, and these partition the set of activities [156, Property 3.1].

The Alpha algorithm [156, Defn. 4.3] (Algorithm 1) derives an SWF-Net $\alpha(\mathcal{W})$ from an event log $\mathcal{W}$. First, three sets of activities are created. $T_{\mathcal{W}}$ is the set of transitions in the net, containing all unique activities from $\mathcal{A}$ that occur in $\mathcal{W}$. T_I and T_O contain the 'input' and 'output' transitions in the net, activities that appear first and last in traces in $\mathcal{W}$, respectively (obtained by auxiliary functions $first(x), last(x)$). In step 2 sets $R_>, R_\rightarrow, R_\parallel, R_\#$ are created. Elements of $R_>$ are pairs of activities related by the $>$ relation, and similarly for the other relations.

Places $P_{\mathcal{W}}$ in the net are defined in two stages. First (steps 3–18), $X_{\mathcal{W}}$ is created as the set of pairs (A, B) of sets of transitions for which each activity b in B is a successor of each activity a in A, i.e. $a \rightarrow b$, and all activities in A are related by the $\#$ relation, as are activities in B. Thus,

$$X_{\mathcal{W}} = \{(A, B) \in \{\mathcal{P}(T_{\mathcal{W}}) \times \mathcal{P}(T_{\mathcal{W}})\} \mid \forall_{a \in A} \forall_{b \in B}\, a \rightarrow b \wedge \forall_{a_1, a_2 \in A}\, a_1 \# a_2 \wedge \forall_{b_1, b_2 \in B}\, b_1 \# b_2\},$$

where $\mathcal{P}(T_{\mathcal{W}})$ is the power set of $T_{\mathcal{W}}$ (the set of all subsets of $T_{\mathcal{W}}$).

$X_{\mathcal{W}}$ is created by iterating over all possible pairs of activities $(a, b) \in T_{\mathcal{W}} \times T_{\mathcal{W}}$. If two such activities a, b are related by $a \rightarrow b$, a pair of sets (A, B) is created; $A = \{a\}, B = \{b\}$, and $T_{\mathcal{W}}$ is then iterated over again (steps 8–15). Each activity that correctly relates to

Algorithm 1 Alpha Process Mining Algorithm (see [156, Property 3.1])

Input: $\mathcal{W}$, an event log over activities $\mathcal{A}$.
Output: an SWF-Net $\alpha(\mathcal{W}) = (P_\mathcal{W}, T_\mathcal{W}, F_\mathcal{W})$.

1: $T_\mathcal{W} \leftarrow \{a \in \mathcal{A} \mid \exists_{x \in \mathcal{W}}\, a \in x\}$.
$T_I \leftarrow \{a \in \mathcal{A} \mid \exists_{x \in \mathcal{W}}\, a = first(x)\}$.
$T_O \leftarrow \{a \in \mathcal{A} \mid \exists_{x \in \mathcal{W}}\, a = last(x)\}$.
(Functions $first(x)$, $last(x)$ return the first and last activities, *resp.*, of trace x.)
2: $R_> \leftarrow \{(a,b) \in \mathcal{A} \mid a > b\}$, $R_\rightarrow \leftarrow \{(a,b) \in \mathcal{A} \mid a \rightarrow b\}$,
$R_\| \leftarrow \{(a,b) \in \mathcal{A} \mid a \parallel b\}$, $R_\# \leftarrow \{(a,b) \in \mathcal{A} \mid a \# b\}$.
3: $X_\mathcal{W} \leftarrow \emptyset$.
4: **for** $a \in T_\mathcal{W}$ **do**
5: **for** $b \in T_\mathcal{W}$ **do**
6: **if** $(a,b) \in R_\rightarrow$ **then** $A \leftarrow \{a\}, B \leftarrow \{b\}$
7: $T'_\mathcal{W} = T_\mathcal{W}$. Add activities to A, B:
8: **for** $c \in T'_\mathcal{W}$ **do**
9: **if** $\forall d \in A, (c,d) \in R_\# \wedge \forall e \in B, (c,e) \in R_\rightarrow$ **then** $A \leftarrow A \cup c$.
10: **end if**
11: **if** $\forall d \in B, (c,d) \in R_\# \wedge \forall e \in A, (e,c) \in R_\rightarrow$ **then** $B \leftarrow B \cup c$.
12: **end if**
13: $T'_\mathcal{W} = T'_\mathcal{W} \setminus c$.
14: **end for**
15: $X_\mathcal{W} \leftarrow X_\mathcal{W} \cup (A, B)$.
16: **end if**
17: **end for**
18: **end for**
19: $Y_\mathcal{W} \leftarrow \{(A,B) \in X_\mathcal{W} \mid \forall_{(A',B') \in X_\mathcal{W}} A \subseteq A' \wedge B \subseteq B' \Longrightarrow (A,B) = (A',B')\}$.
20: $P_\mathcal{W} \leftarrow \{p_{(A,B)} \mid (A,B) \in Y_\mathcal{W}\} \cup \{i_\mathcal{W}, o_\mathcal{W}\}\}$.
21: $F_\mathcal{W} \leftarrow \{(a, p_{(A,B)}) \mid (A,B) \in Y_\mathcal{W} \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A,B) \in Y_\mathcal{W} \wedge b \in B\}$
$\cup \{(i_\mathcal{W}, t) \mid t \in T_I\} \cup \{(t, o_\mathcal{W}) \mid t \in T_O\}$.

Return: $\alpha(\mathcal{W}) = (P_\mathcal{W}, T_\mathcal{W}, F_\mathcal{W})$.

activities in A and B is added to the sets. Finally, the pair of sets (A, B) is added to $X_\mathcal{W}$. Although the complexity of this stage is exponential in the number of activities, typically there are relatively few activities (less than 100) and the complexity does not depend on the size of the log [156]. $Y_\mathcal{W}$ is created by removing from $X_\mathcal{W}$ any pairs (A', B') of sets of activities which are subsumed by another pair of sets (A, B), i.e. $A' \subseteq A$ and $B' \subseteq B$.

In step 20, the places $P_\mathcal{W}$ in the net are identified. For each pair (A, B) of sets of activities in $Y_\mathcal{W}$, a place is created between the activities in A and those in B. $i_\mathcal{W}$ and $o_\mathcal{W}$ are added as artificial unique start and end places. In step 21, $F_\mathcal{W}$ defines the directed arcs of the net: $(a, p_{(A,B)})$ describes an edge from transition a to a place, $(p_{(A,B)}, b)$ an edge from

a place to transition b. The algorithm returns the completed net $\alpha(\mathcal{W}) = (P_{\mathcal{W}}, T_{\mathcal{W}}, F_{\mathcal{W}})$.

For the running example, the following sets will be created,

$$
\begin{aligned}
T_{\mathcal{W}} &= \{i, a, b, c, d, e, f, g, h, o\},\\
T_I &= \{i\},\\
T_O &= \{o\},\\
R_> &= \{(i,a),(a,b),(a,c),(b,d),(b,e),(d,e),(e,d),(d,f),(e,f),(f,g),(f,h),(c,o),\\
&\qquad (g,o),(h,o)\},\\
R_\rightarrow &= \{(i,a),(a,b),(a,c),(b,d),(b,e),(d,f),(e,f),(f,g),(f,h),(c,o),(g,o),(h,o)\},\\
R_\parallel &= \{(d,e)\},\\
R_\# &= \{(i,c),(i,d),(a,d),\ldots\},\\
X_{\mathcal{W}} &= \{(\{i\},\{a\}),(\{a\},\{c,b\}),(\{b\},\{d\}),(\{b\},\{e\}),(\{d\},\{f\}),(\{e\},\{f\}),\\
&\qquad (\{f\},\{g,h\}),(\{c,g,h\},\{o\})\},\\
Y_{\mathcal{W}} &= X_{\mathcal{W}},\\
P_{\mathcal{W}} &= \{p_0 = i_{\mathcal{W}}, p_9 = o_{\mathcal{W}}, p_1 = p_{(\{i\},\{a\})}, p_2 = p_{(\{a\},\{c,b\})}, \ldots, p_8 = p_{(\{c,g,h\},\{o\})}\},\\
F_{\mathcal{W}} &= \{(p_0,i),(i,p_1),(p_1,a),\ldots,(o,p_9)\},\\
\alpha(\mathcal{W}) &= (T_{\mathcal{W}}, P_{\mathcal{W}}, F_{\mathcal{W}}).
\end{aligned}
$$

3.2.2 Heuristics Miner Algorithm

Recall that $N(ab)$ denotes the count of sub-string ab in traces in $\mathcal{W}$. and $P_{HM,n}(S)$ the probability that Heuristics Miner mines process structure S correctly from a random sample (event log) of n traces.

Heuristics Miner [194] (HM) uses such frequencies to calculate a *Dependency Measure* which is an indication of the strength of the causal relation between a pair of activities, The DM is used to determine the arcs, splits and joins in the process model[1]. The

[1]We describe Heuristics Miner (HM) as implemented in ProM 5.2 [170]. HM uses MXML format logs [167], which allow additional attributes to be attached to activities, but does not use such attributes.

UH $\in \{T, F\}$: 'Use All-Activities--Connected' Heuristic	Default true: assume connected graph, each activity except start and end has at least one successor and predecessor.
PO $\in \mathbb{N}_{>0}$: Positive Observations	Default 10: no. of observations of string ab needed to consider adding additional arc $a \rightarrow b$.
DT $\in [0, 1]$: Dependency Threshold	Default 0.9: only add $a \rightarrow b$ if $\mathrm{DM}_{ab} >$ DT.
RTB $\in [0, 1]$: Relative-To-Best threshold	Default 0.05: only add $a \rightarrow b$ if DM_{ab} is within RTB of largest DMs of relations to existing successors of a or predecessors of b.

Table 3.5: Relevant Heuristic Miner Parameters

Dependency Measure (DM) between $a, b \in \Sigma$ is

$$\mathrm{DM}_{ab} = \frac{N(ab) - N(ba)}{N(ab) + N(ba) + 1} \in [-1, 1]. \tag{3.1}$$

Note that $\mathrm{DM}_{ba} = -\mathrm{DM}_{ab}$.

A number of threshold parameters, described in Table 3.5, act with the Dependency Measure to control the detail in the mined model. In this way noisy logs are handled by allowing user control of the size and complexity of the mined model. In this chapter we assume the 'Use All-Activities-Connected' Heuristic is set to true, which ensures the mined model is a connected graph, i.e. all activities other than the unique start and end activity have at least one successor and predecessor.

The algorithm produces a Causal Matrix (CM), visually represented as a Heuristics Net (Section 2.2), a directed graph in which nodes represent activities and arcs represent causal dependencies. Splits are separately annotated as representing exclusive choice (XOR) between subsequent activities, or to parallel paths (AND). Joins are annotated similarly. Figure 3.3 shows the same information compactly, for our running example.

Given a log of traces, the model is constructed in three main steps, described next: create Dependency Matrix, create Dependency Graph, and identify the types of splits and joins (exclusive or parallel). The Causal Matrix encodes the Dependency Graph and split/join types. Algorithm (2) outlines the main processing steps.

1. *Create Dependency Matrix (steps 1–2):* An $|\mathcal{A}| \times |\mathcal{A}|$ matrix M is created, whose

Algorithm 2 Outline of the Heuristics Miner Algorithm [194] (for Processes without Cycles or 'Long-Distance Dependencies').

Input: $\mathcal{W}$, an event log over s unique activities $\mathcal{A}$, parameters AND, RTB, PO and DT.

Output: Causal Matrix $CM(\mathcal{W}) = (V_{\mathcal{W}}, E_{\mathcal{W}}, I_{\mathcal{W}}, O_{\mathcal{W}})$.

$V_{\mathcal{W}} \leftarrow \{a \in \mathcal{A} \mid \exists_{x \in \mathcal{W}}\, a \in x\}$

$M \leftarrow \begin{pmatrix} M_{11} & \dots & M_{1s} \\ \vdots & \ddots & \vdots \\ M_{s1} & \dots & M_{ss} \end{pmatrix}, M_{ij} = \dfrac{N(a_i a_j) - N(a_j a_i)}{N(a_i a_j) + N(a_j a_i) + 1},$

(where a_i, a_j represent the activities for row i, column j, respectively).

$v_S \leftarrow a_i \in \mathcal{A} \mid \forall j, 1 \leq j \leq s, M_{ij} \leq 0.$

$v_E \leftarrow a_j \in \mathcal{A} \mid \forall i, 1 \leq i \leq s, M_{ij} \leq 0.$

for $1 \leq i \leq s$ **do** ▷ Arcs to successor activities.
    $k = 1$
    for $1 \leq j \leq s$ **do**
        if $M_{ij} > M_{ik}$ **then** $k = j$
        end if
    end for
    $E_{\mathcal{W}} = E_{\mathcal{W}} \cup (v_i, v_k)$.
end for

for $1 \leq j \leq s$ **do** ▷ Additional arcs to predecessors.
    $k = 1$
    for $1 \leq i \leq s$ **do**
        if $M_{ij} > M_{kj}$ **then** $k = j$
        end if
    end for
    $E_{\mathcal{W}} = E_{\mathcal{W}} \cup (v_i, v_k)$.
end for

Add extra arcs to $E_{\mathcal{W}}$ if the conditions in Equations (3.4), (3.5) and (3.6) are met with respect to the RTB, PO and DT parameters.

$I_{\mathcal{W}} : a \rightarrow \{B_1, B_2, \dots B_m\}$, where $a \in V_{\mathcal{W}}, B_i \in \mathcal{P}(V_{\mathcal{W}}), 1 \leq i \leq m,$ and

$$\forall\, (b, b') \in B_i \times B_i, 1 \leq i \leq m, \frac{N(bb') + N(b'b)}{N(ab) + N(ab') + 1} < \text{AND} \wedge$$
$$\forall\, (b, c) \in B_i \times B_j, 1 \leq i < j \leq m, \frac{N(bc) + N(cb)}{N(ab) + N(ac) + 1} \geq \text{AND}\,.$$

$O_{\mathcal{W}} : a \rightarrow \mathcal{P}(\mathcal{P}(V_{\mathcal{W}}))$ is created similarly.

Return: $CM(\mathcal{W}) = (V_{\mathcal{W}}, E_{\mathcal{W}}, I_{\mathcal{W}}, O_{\mathcal{W}})$.

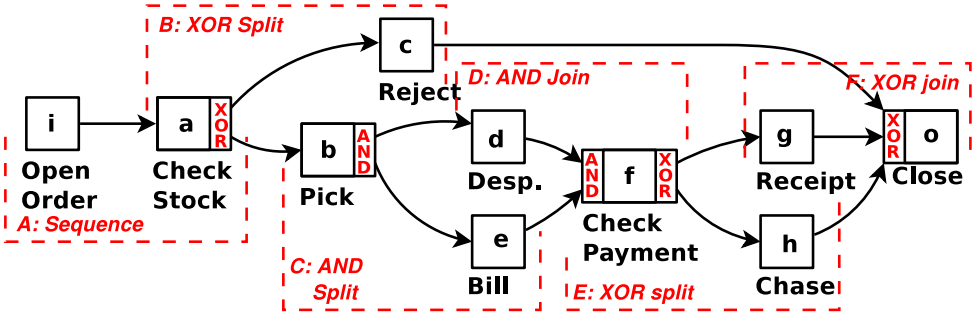


Figure 3.3: Running Example Process for Fulfilling an Order (Model $\mathcal{M}$) as a Directed Graph similar to a Heuristics Net.

elements M_{ij} are the Dependency Measure (DM) between activities a_i and a_j corresponding to row i, column j of M, $1 \leq i, j \leq |\mathcal{A}|$, i.e. $M_{ij} = \text{DM}_{a_i a_j}$. Note that M is antisymmetric ($M = -M^T$), since $M_{ij} = -M_{ji}$ for all such i, j (Equation 3.1)[1].

2. *Create Dependency Graph (steps 3–21):* A node is created for each activity in $\mathcal{A}$ (step 1). A single start (*resp.* end) node v_S (*resp.* v_E) is assumed, the activity for which none of the column (row) entries in M are positive (steps 3–4).

   The graph is connected by identifying a successor and a predecessor activity for each remaining node. This is a two-stage process. First (steps 5–12), activity successors are identified. The successor of an activity a is the activity corresponding to the largest DM in the row for a. Second (steps 13–20), predecessors are identified. The predecessor of activity a is the activity corresponding to the largest DM in the column for a. For example, in Figure 3.3, if $\text{DM}_{ab} \geq \text{DM}_{ac}$ then arc $a \rightarrow b$ will be created[2]. Arc $a \rightarrow c$ will not be created until activity predecessors are identified, when a will be chosen as the only predecessor of b.

   Additional arcs are created (step 21) between two activities a, b if all of the following hold,

   (a) M_{ab} exceeds the DT threshold parameter,

   (b) ab occurs more than PO times in the log, and

   (c) M_{ab} is within RTB of the largest Dependency Measure to existing successors

[1] M^T indicates the transpose of matrix M.
[2] If $\text{DM}_{ab} = \text{DM}_{ac}$ then b or c is selected as the successor of a at random.

of a or predecessors of b.

This is summarised in Equations (3.2)–(3.6). We represent the Dependency Graph through a graph adjacency matrix A, whose elements $A_{ij} = 1$ iff

$$\forall\, k \notin \{i,j\}, M_{ik} < M_{ij} \tag{3.2}$$

$$\vee\, \forall\, k \notin \{i,j\}, M_{kj} < M_{ij} \tag{3.3}$$

$$\vee \Big[(M_{ij} > \mathrm{DT}) \ \wedge \ \big(N(a_i a_j) > \mathrm{PO}\,\big) \wedge \tag{3.4}$$

$$\Big(\exists\, k \notin \{i,j\}, \forall\, l \notin \{i,j,k\}, M_{ik} > M_{il} \ \wedge\ |M_{ij} - M_{ik}| < \mathrm{RTB} \tag{3.5}$$

$$\vee\, \exists\, k \notin \{i,j\}, \forall\, l \notin \{i,j,k\}, M_{kj} > M_{lj} \ \wedge\ |M_{ij} - M_{kj}| < \mathrm{RTB} \Big) \Big], \tag{3.6}$$

$A_{ij} = 0$ otherwise. Terms (3.2) and (3.3) describe the UH parameter and are ignored if this is set to false. The criteria for creating additional arcs are given in terms (3.4), (3.5) and (3.6).

3. *Identify Types of Splits and Joins: (steps 22–23):* The 'AND measure' and associated threshold determine whether a pair of activities b, c following activity a are related exclusively (XOR) or in parallel (AND):

$$a \ \rightarrow\ (b \parallel c) \text{ if } \frac{N(bc) + N(cb)}{N(ab) + N(ac) + 1} > \mathrm{AND}, \quad a \ \rightarrow\ (b \,\#\, c) \text{ otherwise,}$$

In this way the $I_{\mathcal{W}}$ relation relates each activity to a set of sets of its successor activities, e.g. $a \rightarrow \{B, C\}$. Each pair of activities in B are exclusive, as are activities in C, but activities in B are in parallel with those in C. The relation $O_{\mathcal{W}}$ defines joins in a similar manner.

Here we ignore further steps in the Heuristics Miner to identify cycles and 'long-distance' relationships.

In the next two sections we review the main other approaches to process mining, and metrics used for comparing process models.

3.2.3 Other Process Mining Algorithms

There are many approaches to process mining, beyond the two introduced in the previous section. Process mining algorithms can be broadly split into 'local' and 'global' approaches [155]. 'Local' approaches build models from relations between activities, e.g. the Alpha Algorithm [156] (Section 3.2.1), and its extensions Alpha$^+$ [48] which allows 'short loops', and Alpha^{++} [195] which allows 'long-distance' dependencies between activities. These algorithms use pairs of activities in $\mathcal{W}$ to mine a Petri net. They assume that the underlying process can be represented by an SWF-Net (Section 2.1), and that $\mathcal{W}$ contains no noise, i.e. is an exact representation of the underlying process. Extensions to Alpha have been developed to mine processes adhering to specific representations or process structures [199]. Heuristics Miner [194] (Section 3.2.2) and its derivatives similarly use pairs of activities but with softer decision boundaries through (for example) counting occurrences in the event log. 'Global' methods start with and refine a full model, e.g. genetic mining [46, 145] and region mining [20, 164].

Within this local/global split, some algorithms consider different size contexts around an activity ('Extended Relations', Table 3.6). These include approaches which aim to be flexible in dealing with noisy or unstructured processes [164]; allow for 'long-distance' relations between activities [195]; use clustering and abstraction at the level of traces [24,70,137] or activities [73,74,172] to mine processes at different levels of detail; or extract multiple processes from a single event log [71]. Furthermore, while most algorithms build a single model, others build first a model for each type of trace, then aggregate these into a single model, e.g. [54, 141, 171]. Others, rather than specifying the allowed behaviour, use 'declarative languages' to specify the process by the behaviour that is disallowed [96]. Recent work has focussed on efficient mining from large event logs, such as decomposing processes into 'passages' to enable distributed process mining algorithms [160].

In Table 3.6 we summarise the main algorithms with respect to the modelling representations which they use and their inductive biases, or the assumptions which they make. We use the term 'Other Directed Graphs' to refer to simple directed graphs (Section 2.4).

These are differentiated from 'informal' models such as 'Fuzzy Models' [73] and 'Simple Precedence Diagrams' [172] which use nodes to represent either activities or aggregations of activities, varying according to how their parameters are set in interactive use.

The table also notes some of the assumptions made by algorithms to reduce the search space for a mined model. Activities may be treated as atomic (taking no time) or as having duration. Data attributes other than timestamps may be used, for example to cluster activities. Nodes in the model may be required to be uniquely labelled or multiple nodes allowed the same label ('Duplicate Nodes'). All of the detail in the event log may be used, or activities or traces clustered or aggregated to produce a higher-level representation. Only a few algorithms take a probabilistic approach [39, 44, 63, 80]. Finally, we note two specific problems addressed: event logs containing only positive examples [52, 67], by artificially generating negative examples to improve learning ('Negative Events'), and mining from logs lacking case IDs [63] ('Unlabelled Traces').

Some of these approaches to process mining, e.g. Genetic Mining, use process metrics to enable them to converge to their optimal output model. Metrics are also used more widely, to compare process models and assess the quality of models produced by process mining algorithms. We discuss process metrics in the next section.

3.3 Process Mining Metrics

Many methods have been proposed for assessing process mining results and algorithms, and comparing business process models. These are often based on the syntax of the representations used. In the next sections we review generally accepted characteristics of process mining metrics, and summarise existing metrics according to their characteristics.

3.3.1 Characteristics of Process Mining Metrics

A distance metric $d(A, B)$ between entities A, B in a metric space is characterised by [132]

	Representation							Inductive Bias												
Algorithm/Author	Automata	Other Directed Graphs	Probabilistic	Type of Petri net	Informal	Process Languages	Other Representation	Local Relations	Extended Relations	Global Relations	Atomic Events	Timed Events	Extra Attributes	Activity Aggregation	Trace Aggregation	Negative Events	Multiple Processes	Aggregate Trace Models	Probabilistic	Duplicate Nodes	Unlabelled Traces
Agrawal [7]		✓						✓				✓									
Datta [44]	✓								✓			✓							✓		
InWoLvE [78–80]						✓		✓			✓									✓	
Herbst HMM [80]			✓					✓			✓			✓					✓	✓	
RNet [39]	✓									✓	✓			✓							
KTail [39]	✓									✓	✓			✓							
Markov [39]	✓									✓	✓			✓					✓		
Schimm Block-Structured [133]						✓		✓				✓									
Alpha Algorithm [156]				✓				✓			✓										
Alpha$^+$ [48]				✓				✓			✓										
Interval Sorted [113]						✓		✓				✓						✓			
Multi-Phase Miner [54, 171]						✓		✓			✓							✓		✓	
Heuristics Miner [194]		✓						✓			✓										
Two-Step Approach [164]	✓			✓					✓		✓		✓								
Genetic Miner [46, 47, 49, 50, 151]		✓								✓	✓										
Hierarchy Discovery [51, 70, 71]		✓								✓	✓				✓		✓				
Alpha^{++} [195]				✓					✓		✓										
Alpha#,γ, τ, γ^+ [199]				✓				✓			✓										
Beta [117]				✓				✓				✓									
Region Miner [20, 55]				✓						✓	✓										
Fuzzy Miner [27, 73, 74, 91, 163]					✓			✓				✓	✓	✓							
Genetic Programming [145]		✓								✓	✓										
Simulated Annealing [138]		✓								✓	✓										
Trace Clustering [24, 25, 137]				✓			✓	✓			✓				✓						
Visualisation [172]					✓				✓		✓			✓							
Expectation-Maximisation [63]			✓							✓	✓								✓		✓
Agnes [67]				✓						✓	✓					✓					
WorkFlowMiner [65]	✓			✓			✓	✓				✓									
S Algorithm [85]				✓				✓			✓										
Heuristics Miner^{++} [28, 29]		✓						✓				✓									
Flexible Heuristics Miner [193]		✓						✓			✓										
DPML, Alchemy [19, 97]							✓			✓	✓										
Declare [97]							✓			✓	✓										
Aperture [141]		✓						✓				✓	✓				✓				

Table 3.6: Process Discovery Algorithms, in approximate Chronological Order.

- *non-negativity*: $d(A, B) \geq 0$,
- *symmetry*: $d(A, B) = d(B, A)$,
- *identity of indiscernibles*: $d(A, B) = 0$ iff A = B, and the
- *triangle inequality*: $d(A, C) \leq d(A, B) + d(B, C)$.

Becker and Laue [18] apply these to distance metrics for comparing process models (for searching a repository of processes) and add the following desirable characteristics:

1. take into account commonality between models as well as differences, e.g. the number of different nodes relative to the total number of nodes;
2. take account of similarity between activity names;
3. impose no specific restrictions on structure (such as disallowing cycles); and
4. be efficient to calculate.

More generally, Rozinat *et al.* [127] define desirable characteristics of a metric for process mining as

1. *validity*: the measure should be correlated with the property being measured, e.g. as the difference between models increases, so does the metric;
2. *stability*: the metric should be affected as little as possible by properties other than that being measured;
3. *analysability*: values must 'make sense', for example varying intuitively between optimal best and worst values;
4. *reproducibility*: between the same two models, the metric must always evaluate to the same result; and
5. *localisability*: the method of calculating the metric should enable differences to be related to locations in the model structure.

Four *dimensions* [122] are commonly used for describing comparisons between process models and between models and event logs:

- *fitness*: how much of the event log could have been produced by the mined model (also known as *recall*);

- *behavioural appropriateness*: the opposite, i.e. how much extra behaviour the model allows, for which there is no evidence in the event log (also known as *precision*);
- *structural appropriateness*: notions of the simplicity or efficiency of the structure of the model; and
- *generalisation*: how well the model fits 'extra behaviour' likely to exist in the underlying model, which was not seen in the samples seen in the event log.

There are tensions between these dimensions: A more general model allows extra behaviour not seen in the log, and so is less behaviourally appropriate. Similarly, fitness and behavioural appropriateness are in tension, while a simpler structure may either restrict or extend the set of traces supported by the model, depending on the representation.

3.3.2 Conformance Metrics from the Process Mining Literature

We briefly review metrics from the process mining literature, along these dimensions.

Fitness Metrics: Completeness [70], Naïve Behavioural Recall^B [12, 168] and Parsing Measure [194] measure the proportion of traces in a log, which are supported by the mined model. Weijters *et al.* [194] argue that this over-penalises the model and that partially-fitting traces should be penalised less heavily. Therefore the Continuous Parsing Measure (CPM) [194], reduces the granularity of comparison to the activity level. Token-Based Fitness (f) [124, 127] refines CPM in the context of 'replaying' traces through a Petri net, penalising for tokens artificially created to 'force' the parsing, or remaining after parsing. f takes account of frequencies of traces in the log, but it is affected by the structure of the net, and penalises a series of missing activities no more than just one.

Literature on Genetic Process Mining [46, 47, 49, 145, 151] refines f to allow varying 'punishment' of different types of error, to attempt to construct improved fitness functions for the evolutionary evaluation of fitness. These measures are most commonly known as $PF_{complete}$, although this name is also used for a different metric at the trace level [66] and for a metric combining recall and precision [181]. An enhanced version of f [12, 46, 168]

allows comparison of a model with 'observed behaviour' in a 'reference' event log, possibly different from the log from which the model was mined. The idea is that the reference log is in some sense a 'ground truth' (large sample), approximating the true underlying behaviour. The distance is scaled by frequency and length of traces, and size of the log.

Other 'recall' metrics include: r_B^p [67] which measures as 'false negatives' just the part of f where transitions were forced to fire to parse the log; Event Coverage c_E [124] which counts the event labels (in the log) which are also activity labels (in the model); and a simple structural metric [113] which counts arcs missing from a mined model compared with a target model.

Precision/Generalisation: Soundness [70] and Naïve Behavioural Precision^B [12,168] measure the proportion of the traces which could be generated by the model, for which there is evidence in the log. The models to which these are applied are acyclic, so the behaviour is finite. Behavioural Appropriateness a_B is first defined [124,127] as the ratio of number of Petri net transitions enabled as traces are replayed, to number of activities in the model. This is motivated by the assumption that the more transitions that are enabled, the more extra behaviour the model is likely to support. a_B takes account of trace frequencies, but can measure trace-equivalent models differently and over-penalises parallelism. Improved [127] and Advanced [124] Behavioural Appropriateness a'_B refine the measure to the activity level by comparing activities which sometimes follow or precede each other in the model, and in the log. These metrics are computationally costly, requiring exploration of the state space of the model: practical implementations employ heuristics to limit this exploration. Behavioural Specificity s_B^n [67] avoids this problem since its associated process mining method (Agnes) artificially generates 'negative examples' and calculates the measure during replay of these traces with negative examples.

Other Behavioural Precision metrics [12,46,168] penalise extra behaviour allowed by a model compared with a reference log, scaling for the lengths and frequency of traces, and the size of the log. Various measures called $PF_{precise}$ are defined, in [46,66,145] as weaker

measures related to those already described, and in [181] as a unified recall and precision metric for Genetic Process Mining. ETC Precision takes a different approach [106]. It aims to efficiently estimate the effort to correct (compared with the log) the mined (Petri net) model, and to locate discrepancies in the model. A transition system TS_M is built from the model and assumed to be bigger (more states and transitions) then TS_L built from the log, so transitions from TS_M which 'escape' from TS_L are counted. Model Coverage c_T [124] measures how many labels of visible activities in the model are also event labels in the log. A simple structural metric [113] counts arcs in the mined model, which are not in a target model.

Structural Metrics: These attempt to reward 'simple' or otherwise efficient or coherent model structures, such as those using 'workflow patterns' [154]. Several measures compare two models, i.e. a mined and a reference model: $precision^S$ and $recall^S$ compare the numbers of connections between transitions in two Petri nets, Structural Precision S_P and Recall S_R [46] similarly compare 'Causal Matrices'. Duplicates Precision D_P and Recall D_R are similar but take account of duplicately labelled activities. A similar measure is used in [113]. These measures however are able to assign a high similarity to models which are in fact structurally very different, and do not take account of model semantics such as whether splits are exclusive (XOR) or parallel (AND).

Other measures apply to a single model. Improved [127] or Advanced [124] Structural Appropriateness a'_S applies to Petri nets and penalises duplicately labelled activities which do not occur together in a trace, and redundant 'invisible' transitions. Lassen *et al.* [90] discuss metrics for process models generally (rather than specifically for process mining). The Cardoso Metric allocates penalties to different types of split, and the Extended Cardoso Metric applies this to Petri nets. They argue however that the penalties applied do not correspond well to the behaviour of the model and introduce the Extended Cyclomatic Metric to measure the 'connectedness' of the reachability graph of a Petri net. Finally the Structuredness Metric is introduced. It is calculated algorithmically by

decomposing a Petri net into well-defined components, assigning weights to the steps in the process. This is used [181] as part of a genetic fitness measure $PF_{uncomplex}$.

Combined Metrics: In the Genetic Process Mining literature [46, 66, 145, 181] several measures are combined and weighted to produce flexible genetic fitness functions. Ferreira and Gillblad [63] define G-score (in fact the Bhattacharyya Coefficient [21]) as a similarity measure between processes described as probabilistic automata, which in effect combines recall and precision. Dependencies between activities in two models are compared in [29], defining the F_1 measure as the 'harmonic mean between precision and recall'.

Recent Work: Process mining conformance remains an area of active research. Recent work is motivated by a move towards more flexible processes (*cf* Adaptive Case Management [140]) and related representations. Fitness measures associated with 'semi-flexible' models [5] are defined, with a modified A^* search algorithm to fit (partial) traces to the model. Related work [6] allows costs to be allocated to skipping or inserting extra activities. Behavioural profiles [192] attempt a more intuitive and flexible measure of the precision of a model, by comparing the relations between pairs of activities. These partially take into account that the log is only a sample of behaviour, by defining some relations (e.g. sequential) to 'subsume' others (e.g. parallel). The view is taken in [60] that processes are 'artefact centric' (described by the interaction of objects) and may split into and join sub-processes, known as 'proclets', for example to split an order between suppliers and customers. Related measures of conformance are defined.

3.3.3 Metrics for Querying Business Process Repositories

The wider business process literature also introduces metrics for comparing process models, for example to search Business Process repositories for a specific model, or models which fully or partially match some criteria such as structural fragments or data attributes. The problems of determining isomorphism of two graphs, and calculating graph edit dis-

tance between them, are thought to be of NP-Complete complexity (see e.g. [62, 102]), i.e. without known solutions of polynomial complexity. Approximate matching methods for very large graphs have been developed for bioinformatics (see e.g. [198]). However, graphs for business processes tend to be relatively smaller and more structured [133, 154].

Bae *et al.* [15, 16] compare the structures of two directed process graphs using their graph adjacency matrices to calculate the sum of squares of arc differences. This is extended in [190] to allow for weighted arcs and data attributes associated with nodes. Van Dongen *et al.* [173, 174] investigate efficient storage and retrieval of process models. They define Causal Footprints to describe the behaviour of processes in terms of the sets of activities which may follow or precede each activity. Causal Footprints are represented as vectors and compared by their inner product. They also present methods comparing activities by string comparison of their names, and based on their neighbouring nodes in the model. These methods are extended [53] to compare models by node similarity, structural similarity (graph edit distance), and behavioural similarity (Causal Footprints). The methods are described for 'Business Process Graphs', designed as a superset of other representations.

Uba *et al.* [146] aim to detect similar processes, rather than calculating distances. Using a method due to Vanhatalo *et al.* [176] models are deconstructed to a *Refined Process Structure Tree* (RPST) of component structures. The problem of detecting isomorphic graphs is simplified using characteristics of these structures, and representing graphs as unique strings. Other methods include comparing processes by the Principal Transition Sequences which they support [182], broken into full sequences, sequences before and after cycles, and cycles. This method aims to be structurally-independent, and efficiently deal with models with infinite state spaces. The Transition Adjacency Relation (TAR) presented by Zha *et al.* [200, 201] is described as a 'genetic footprint' of a (Petri net) model and as a true metric. Efficient methods for calculating the TAR are given, applied to clustering process models. A similar method in [8] is based on deconstructing a *sound* Petri net using the Alpha algorithm [156] relations.

		Log	Model	Reference Model	Automata	Directed Graph	Probabilistic	Type of Petri net	Flexible Model	Unit of Behaviour	Syntactic	Semantic	Probabilistic
Fitness Metrics	Completeness [70]	✓	✓			✓				trace	✓		
	Naïve Behavioural Recall^B [12, 168]	✓	✓					✓		trace	✓		
	Parsing Measure [194]	✓	✓			✓				trace		✓	
	Continuous Parsing Measure (CPM) [194]	✓	✓			✓				activity		✓	
	Token-Based Fitness (f) [124, 127]	✓	✓					✓		activity		✓	
	$PF_{complete}$ [46, 47, 49, 145, 151]	✓	✓					✓		activity		✓	
	$PF_{complete}$ [66], [181]	✓	✓					✓		activity		✓	
	f [12, 46, 168]		✓	✓				✓		activity		✓	
	Behavioural Recall r_B^p [67]	✓	✓					✓		activity		✓	✓
	Log Coverage C_E, C_{LE} [124]	✓	✓					✓		activity	✓		
	Simple structural metric [113]		✓	✓		✓				node/arc	✓		
Precision/Generalisation	Soundness [70]	✓	✓			✓				trace	✓		
	Naïve Behavioural Precision^B [12, 168]	✓	✓					✓		trace	✓		
	Behavioural Appropriateness a_B [124, 127]	✓	✓					✓		activity	✓		
	Improved Beh. Approp. a'_B [127]	✓	✓					✓		activity	✓		
	Advanced Beh. Approp. a'_B [124]	✓	✓					✓		activity	✓		
	Behavioural Specificity s_B^n [67]	✓	✓					✓		activity		✓	✓
	$PF_{precise}$ [46, 66, 145]	✓	✓					✓		activity		✓	
	$PF_{precise}$ [181]	✓	✓					✓		activity		✓	
	ETC Precision [106]	✓	✓					✓		activity/arc	✓		
	Model Coverage C_T, C_{LT} [124]	✓	✓					✓		activity	✓		
	Simple structural metric [113]		✓	✓		✓				node/arc	✓		

Table 3.7: Fitness (Precision) and Recall Process Metrics

		Log	Model	Reference Model	Automata	Directed Graph	Probabilistic	Type of Petri net	Flexible Model	Unit of Behaviour	Syntactic	Semantic	Probabilistic
Combined Metrics	Genetic [46, 66, 145, 181]		✓	✓		✓				activity	✓	✓	
	G-score [63]	✓	✓		✓		✓			trace			✓
	F_1 measure [29]	✓	✓			✓				activity	✓		
	Dependency Difference d [15, 16]		✓	✓		✓				node/arc	✓		
	Causal Footprints [53, 173, 174]		✓	✓		✓		✓		activity	✓		
	Principal Transition Sequences [182])		✓	✓				✓		part traces	✓		
	TAR [200, 201]		✓	✓				✓		transitions	✓		
Structural Metrics	$precision^S$ [12, 168]		✓	✓				✓		transition pair	✓		
	$recall^S$ [12, 168]		✓	✓				✓		transition pair	✓		
	Structural Precision S_P [46]		✓	✓		✓				arc	✓		
	Structural Recall S_R [46]		✓	✓		✓				arc	✓		
	Duplicates Precision D_P [46]		✓	✓		✓				arc	✓		
	Duplicates Recall D_R [46]		✓	✓		✓				arc	✓		
	Simple structural metric [113]		✓	✓		✓				node/arc	✓		
	Improved Struct. Approp. a'_S [127]		✓					✓		transition/activity	✓		
	Advanced Struct. Approp. a'_S [124]		✓					✓		transition/activity	✓		
	Cardoso Metric [90]		✓					✓		sub-structure	✓		
	Extended Cardoso Metric [90]		✓					✓		sub-structure	✓		
	Extended Cyclomatic Metric [90, 181]		✓					✓		sub-structure	✓		
Flexible Model Metrics	Flexible-Model Metrics [5]	✓	✓						✓	activity		✓	
	Align-Based Precision Metric A^1_P [4]	✓	✓						✓	activity		✓	
	Cost-Based f [6]	✓	✓						✓	activity		✓	

Table 3.8: Combined Precision and Recall, and Structural Process Metrics, and Metrics for Flexible Models

In Tables 3.7 and 3.8 we summarise the main process mining metrics by the dimension which they measure, and by their characteristics. The table indicates which metrics compare an event log with a process model, and which compare two models; the representation on which they operate; and the unit of behaviour which is measured, i.e. traces or activities, or nodes/arcs in the representation. The final three columns show whether the metric is tied to syntactic elements of the representation (e.g. presence of nodes or arcs), semantic elements (e.g. 'playing the token game' in a Petri net), or are in any way probabilistic. Only very few metrics take a probabilistic view; the G-score [63] which compares the log and model as probabilistic automata, and the Behavioural Specificity s_B^n and Recall r_B^p [67], which use the semantic behaviour of the Petri net but are also dependent on the probability of adding artificial negative events to the event log.

3.4 The Need for a Framework for Process Mining

The fundamental problem motivated by the discussion in the preceding sections is that there are many algorithms, process representations and metrics, which do not admit direct comparison, and no unifying representation or framework within which process models and process mining algorithms may be analysed and compared.

Parts of this problem have been considered. In [122] an 'architectural' framework is proposed. This would be a repository for algorithms, methods for comparison, and example event logs and process mining tools. Existing metrics are reviewed and compared with a method of assessing algorithms using k-fold cross-validation. Metrics are found to have the advantage of allowing different aspects of models' behaviour to be compared, and differences localised in the representation in use, but do not provide a common basis for comparing models in different representations, or upon which to objectively discuss other process mining tasks such as generalisation, clustering or abstraction. The k-fold approach uses an experimental method from machine learning, and allows the significance of differences to be quantified. However, it does not provide a theoretical foundation for

analysing the learning behaviour of algorithms and predicting how much data is needed for mining. The paper concludes that more research is needed. Process metrics are also reviewed in [144], which concludes that 'more research is required to enable the production of a generic framework for the quantified comparison of processes'.

The problem of multiple process representations, particularly the differences between languages with formal semantics, favoured in the academic community, and less formal representations preferred by the business community, are discussed in [95]. Methods are reviewed for translating various representations to Petri nets as a unifying representation. Business Process Graphs (BPGs) are introduced in [53] as a generic representation which can capture the details of other model representations, although no methods are given for transforming models in other representations into BPGs. Neither of these models allow for a probabilistic view of processes.

Researchers have begun to consider the question of how much data is needed for mining. Bose *et al.* [26] highlight that this question has 'hardly yet been addressed', noting that 'it would be interesting to know the lower bound on the number of traces required to discover a process model with a desired fitness'. The effect of an algorithm's inductive bias on the 'completeness' of the log is discussed in [148, 156]. For example, when activities can occur in parallel, then the number of possible sequences of activities (traces) which can appear in $\mathcal{W}$ increases exponentially with the number of parallel activities. When cycles are allowed, the number of possible traces is infinite. To mitigate this, the Alpha algorithm, rather than needing to see all possible traces, requires only that any pair of activities that can occur in succession, must appear in $\mathcal{W}$. However, these discussions do not take into account the probability of these sequences appearing in $\mathcal{W}$.

Yang *et al.* [197], and van Hee *et al.* [175] do consider this question probabilistically. In [197], the Chebyshev inequality, which bounds the likely deviation of observed samples from a distribution from their expected values, is used to discuss bounds on the length of the log used for mining, to ensure all possible traces are included, and for trace frequencies to approximate the probabilities in the underlying distribution. These bounds are very

loose since they take no account of the behaviour of the mining algorithms. In [175], similar methods are applied to the Alpha algorithm, using Hoeffding bounds and hypothesis tests on the probabilities of seeing each required pair of activities in the log. The method proposed in this thesis is more general, and can be used to derive closer bounds for specific algorithms and models.

Some early process mining approaches made use of probabilistic concepts to produce non-probabilistic models [44], discover concurrency [38], or estimate some probabilities in the model (e.g. [78, 79]). These approaches did not take a fully probabilistic view of process mining such as we present in this thesis. Our work has more in common with that in [63], which considers a stream of symbols representing activities, produced by multiple random sources, and uses an Expectation-Maximisation procedure to construct a type of stochastic automaton most likely to represent the underlying process. We rather consider a single source generating traces.

We conclude that this is an important problem, which we address in the next chapters.

3.5 Chapter Summary

In this chapter we reviewed business process mining and described the problem addressed by this thesis. Our main question is how to think about process mining algorithms and process models, in a way which allows different algorithms and their results to be compared objectively, and provides a rigorous basis for addressing process mining questions of interest, such as simplifying or generalising models, dealing with noisy data, or simulating changes to the process. We introduced two common process mining algorithms in detail, the Alpha Algorithm [156] and Heuristics Miner [194]. We also reviewed the other main mining algorithms and process metrics used to compare the models which they produce.

In the next chapter we present a probabilistic framework for considering process mining algorithms, which allows this type of analysis and comparison. In subsequent chapters we apply the framework to the analysis of two common algorithms, and to the investigation of two practical applications.

CHAPTER 4

A FRAMEWORK FOR THE ANALYSIS OF PROCESS MINING ALGORITHMS

Because of the diversity of process mining algorithms and representation languages for business processes, methods are needed for analysing the behaviour of algorithms. In this chapter we introduce a framework for one such method. Given a probability and a process mining algorithm, how much data of a given, finite process do we need to, with a stated probability, produce a business process 'close enough' to the original? There are two main prerequisites to answering this question. Firstly, a unifying view of processes to allow objective, language-independent analysis, and secondly a notion of 'closeness' to evaluate how similar two processes are.

To satisfy these requirements, we consider the control-flow of business processes as probability distributions over traces of activities, and mining algorithms in terms of their ability to learn such distributions. We use probabilistic automata (e.g. PDFA [179]) as a unifying representation, as these represent a large class of probability distributions over sequences, and act as a lowest common denominator to which to convert models in other languages. The distance between processes can be calculated from PDFA with various metrics. In this chapter we use the d_2 distance, and metrics based on the Bhattacharyya Coefficient [21, 37] and on the Kullback-Leibler Divergence. Under this view, the main task of a process discovery algorithm is then to learn such distributions from data: a machine learning problem.

We begin in Section 4.1 by describing and motivating our probabilistic view of business processes and machine learning view of process mining. Then in Section 4.2 we introduce and describe our framework. We show how a process model can be broken down into sub-structures and the probability of correct mining of those sub-structures, and thus of the full model, accurately calculated. The probabilistic view is supported in Section 4.3 by a comparison of the distances which we use to compare processes, with existing metrics. Much of the material in this chapter was first presented in [184].

In subsequent chapters we will validate the framework presented in this chapter by applying it to the analysis of two process mining algorithms and to two practical problems of current interest in the process mining field.

4.1 Business Processes as Distributions over Traces

In this section we describe our probabilistic view of business processes as distributions over traces, using the notation introduced in Table 3.3. We show how this view enables comparison of processes using distances between probability distributions, and using statistical and hypothesis tests, and motivates a machine learning view of process mining.

While early process mining approaches made use of probabilistic concepts to produce non-probabilistic models (e.g. [38, 44]), or estimated some probabilities in the model (e.g. [78]), we view the control-flow of a business process in an abstract sense as a distribution over strings of symbols representing process traces (see [184]). This has more in common with the work of [63].

We make some restrictions to simplify the analysis, equivalent to those used elsewhere in the literature, e.g. [7, 39, 156]. A process has a single input (start) activity (or task), labelled i, and a single output (end) activity labelled o. The events of activities' occurrence are recorded as they occur, and events are atomic (take no time) and are uniquely labelled, the same label always referring to the same event, and vice versa. No use is made of additional information (such as timing) about events, merely the order in which they are

recorded. The underlying process model is assumed to be fixed. This is unlikely in reality, but we assume that any change is slow enough to be ignored over the period that data is collected. Also, we do not consider cycles.

We view the control-flow of business processes as probability distributions over strings of symbols $a \in \Sigma$ representing activities (Table 3.3). In other words, a business process can be considered a stochastic regular language $\mathcal{M}$ that describes the probability distribution P_M over Σ^+ (the set of all non-empty strings of activities), such that

$$\sum_{x \in \Sigma^+} P_M(x) = 1. \tag{4.1}$$

$P_M(x)$ therefore gives the probability that if a single trace is drawn at random from process $\mathcal{M}$, that trace will be equal to x.

We define the set of *valid* process traces $\mathcal{T} \subset \Sigma^+$. Since we are not considering cycles, $\mathcal{T}$ is finite, each valid trace $x \in \mathcal{T}$ is finite in length, and no activity $a \in \Sigma$ occurs more than once in x, In addition, each trace begins with the start activity i and finishes with end activity o, and $x = iwo$, $w \in \{\Sigma \setminus \{i, o\}\}^*$ (w may be empty). Note that

$$x \notin \mathcal{T} \Rightarrow P_M(x) = 0, \text{ but } P_M(x) = 0 \nRightarrow x \notin \mathcal{T}.$$

If trace x is not in the set of valid traces $\mathcal{T}$, then it cannot be supported by the process model $\mathcal{M}$. However, if x has probability zero under distribution P_M describing $\mathcal{M}$, this does not mean it is not a valid process trace (e.g. in some other process model).

Given an underlying source $\mathcal{M}$, let $P_M(a|y)$ denote the probability that after seeing the sequence of activities given by string $y \in \Sigma^+$, the next symbol to be seen will be $a \in \Sigma$. This is given by the total probability of all strings with prefix ya (Table 3.3), conditioned on the probability of all strings with prefix y:

$$P_M(a|y) = \frac{P_M(ya\Sigma^*)}{P_M(y\Sigma^*)}.$$

This extends naturally to sub-strings $z \in \Sigma^n$:

$$P_M(z|y) = \frac{P_M(yz\Sigma^*)}{P_M(y\Sigma^*)}, \text{ where } \sum_{z \in \Sigma^n} P_M(z|y) = 1.$$

We also introduce some shorthand notation: We write $\pi(ab)$ for $P_M(i\Sigma^*ab\Sigma^*o)$, the probability of ab occurring in a trace, i.e. the sum of probabilities of all strings beginning with i, ending with o, and containing ab. Similarly, we write $\pi(b|{\rightarrow}a)$ for $P_M(b|i\Sigma^*a)$, the conditional probability that given that a occurs in a trace, the next symbol will be b.

Consider event log $\mathcal{W}_n$ of n traces drawn from $\mathcal{M}$. Since we assume no cycles, a sub-string $w \in \Sigma^+$ occurs zero or one times in any trace in $\mathcal{W}_n$, with probability $\pi(w)$. Then $N(w)$, the number of times w occurs in event log $\mathcal{W}_n$, is Binomially distributed,

$$Q_n\big(N(w)\big) = Bin\big(\pi(w), n\big).$$

This extends to full traces, $N(x)$ is Binomially distributed for each full trace $x \in \mathcal{W}_n$, with probability parameter $\pi(x)$. If $\mathcal{W}_n$ contains m unique traces (i.e. excluding duplicates), then the joint distribution of the counts $N(x_i)$ of the unique traces x_i in $\mathcal{W}_n$, $1 \leq i \leq m$, are described by a multinomial distribution.

The notation is summarised in Table 3.3.

In this thesis, such distributions will be described using transition-labelled *Probabilistic Deterministic Finite Automata* (PDFA) (Section 2.3, *cf* DPFA [179], PDFA [57], DSFA [33]). See also [63, 79]. PDFA provide a 'common denominator' to which processes in other modelling languages can be converted and analysed,

4.1.1 Distances between Probability Measures

Viewing business processes as probability distributions, we can quantify differences between two business processes P_1 and P_2 (e.g. the 'ground truth' and its inferred proxy) via distances on the space of distributions over traces, e.g.

Euclidean Distance

$$d_2(P_1, P_2) = \sqrt{\sum_x \big(P_1(x) - P_2(x)\big)^2},$$

Bhattacharyya Distance [37]

$$d_{Bhat}(P_1, P_2) = \sqrt{1 - \sum_x \sqrt{P_1(x)P_2(x)}},$$

Kullback-Leibler Divergence [89]

$$d_{KL}(P_1, P_2) = \sum_x P_1(x) \log \frac{P_1(x)}{P_2(x)}.$$

Note that Kullback-Leibler Divergence is not a distance measure since it is not symmetric. Also it requires P_1 and P_2 to have the same support. This is straightforward to work around, e.g. by postulating the *Jensen-Shannon Divergence* [93]

$$d_{JSD}(P_1, P_2) = d_{KL}(P_1, \psi) + d_{KL}(P_2, \psi), \tag{4.2}$$

where $\psi(x) = \frac{1}{2}\big(P_1(x) + P_2(x)\big)$.

4.1.2 Statistical Tests on PDFA and Event Logs

Our probabilistic view of business processes also allows statistical tests between an event log and a process represented as a distribution. Consider event log $\mathcal{W}$ of n traces drawn from an unknown process, and a reference process distribution P_M. Let $m = |\mathcal{T}|$ be the number of valid traces supported by P_M. Since P_M has finite support (as we do not consider cycles), and we assume traces to be drawn *i.i.d.*, Pearson's *Chi Square Test*

[110, 136] for example can be used to test whether $\mathcal{W}$ was sampled from P_M:

$$\chi_s^2 = \sum_{i=1}^{m} \frac{\big(N(x) - nP_M(x)\big)^2}{nP_M(x)}, \text{ and}$$
$$p = Pr(\chi_{m-1}^2 \geq \chi_s^2) = \int_{\chi_{m-1}^2=\chi_s^2}^{\infty} f(\chi_{m-1}^2)\, d(\chi_{m-1}^2).$$

$N(x)$ is the number of times x occurs in $\mathcal{W}$, χ_s^2 the sample Chi^2 statistic, and $f(\chi_{m-1}^2)$ the density function of the Chi^2 distribution with $m-1$ degrees of freedom. $N(x)$ is Binomially distributed with mean $nP_M(x)$, and for large enough n, approximately Normally distributed. Since the differences $N(x) - nP_M(x)$, will also be approximately Normally distributed, their squares will follow a Chi^2 distribution. p (the 'p-value') gives the probability that the Chi^2 distribution exceeds the measured value. A low p (typically $p \leq 0.05$) indicates a more statistically significant result, i.e. that $\mathcal{W}$ was not drawn from P_M.

4.1.3 Hypothesis Tests on PDFA and Event Logs

We can also compare the structure of PDFA rather than the distributions which they describe. Consider two PDFA A_1, A_2, differing only in arc probabilities. Let $A_1 = (Q, \Sigma, \delta_1, q_0, q_F)$ represent the reference process distribution P_M, and $A_2 = (Q, \Sigma, \delta_2, q_0, q_F)$ be mined from event log $\mathcal{W}$. Let $q \xrightarrow{a} q'$ denote a single arc in A_2, from state q to q', $(q, q' \in Q)$, labelled with $a \in \Sigma$. For each trace $x \in \mathcal{W}$, there is a unique state sequence $Q^{(x)}$ as A_2 parses x (Section 2.3) and $N_2(q)$ traces from $\mathcal{W}$ will visit q as the event log is parsed by A_2.

$$N_2(q) = |\{x \in \mathcal{W} | q \in Q^{(x)}\}|.$$

Let $N_2(q, a)$ be the number of times arc $q \xrightarrow{a} q'$ is followed, such that

$$N_2(q, a) = N_2(q)\delta_2(q, a, q').$$

Following [83] we model $N_2(q,a)$ as a Binomial random variable X, and test the null hypothesis that X is distributed Binomially according to the probability of the arc in A_1, i.e. $X \sim Bin\big(\delta_1(q,a,q'), N_2(q)\big)$. Then if the probability is low under this distribution, of the observed arc usage count $N_2(q,a)$ differing from its expected value,

$$Pr\big(|X - N_2(q)\delta_1(q,a,q')| \geq |N_2(q,a) - N_2(q)\delta_1(q,a,q')|\big) \leq \frac{p}{2},$$

then with probability $1-p$, A_1 and A_2 represent different distributions, i.e. for $\delta = \delta_1(q,a,q')$,

$$\Big(\sum_{k=0}^{N_2(q,a)} \binom{N_2(q)}{k} \delta^k (1-\delta)^{N_2(q)-k} + \sum_{k=N_2(q,a)}^{N} \binom{N_2(q)}{k} \delta^k (1-\delta)^{N_2(q)-k}\Big) \leq p.$$

An interpretation is that with probability $1-p$, the traces in $\mathcal{W}$ which generated PDFA A_2, were not drawn from P_M which generated A_1. A similar approach can be used to compare the observed frequency of traces in $\mathcal{W}$ with their expected frequencies under P_M. We use these approaches in Chapter 7.

4.1.4 Process Mining: a Machine Learning View

In this section we formalise a machine learning view of process mining, noting that some of these ideas are implicit in other work, e.g. [7, 79]. In particular, in [63] a stream of symbols representing activities is produced by multiple random sources. We rather consider a single source generating traces.

We first review relevant machine learning concepts from a process mining viewpoint.

Machine Learning

Machine learning algorithms are designed to automatically learn from experience [104]. In particular, an algorithm is said to learn from experience E with respect to some class of tasks T and performance measure P, if performance of tasks in T, measured by P,

improves with experience E [104]. With respect to process mining, typically the task is to recover a process model $\mathcal{M}$ equivalent to the underlying model, and the experience is the presentation of traces of business activities from an event log $\mathcal{W}$. A process mining metric may be used to evaluate $\mathcal{M}$ (Section 3.3).

This can be considered as a search problem, to find the optimal *hypothesis* ($\mathcal{M}$), within some *hypothesis space* which constrains what models may be considered. Process mining algorithms make use of knowledge about the structure of the search space, to restrict and assist the search, to improve efficiency and the quality of mined models [72].

We consider several specific learning activities in the context of process mining:

Density estimation : given examples $X = x_1, x_2, \ldots, x_n$ from an unknown distribution $P(x)$, learn a distribution $P'(x)$ to approximate $P(x)$, such that the distance $d\big(P(x), P'(x)\big)$ between the distributions is minimised, for some notion of distance between probability distributions. While most process mining algorithms produce models in non-probabilistic representations, we argue in this thesis for considering the core process mining activity of discovering models of the *control-flow* structure of processes, as one of density estimation.

Clustering : given examples $X = x_1, x_2, \ldots, x_n$, partition X into k sets such that the sum of distances between examples in each set is minimised, for some measure of distance between examples. Process mining algorithms use clustering techniques to create simpler models with 'representative' nodes that aggregate similar activities (e.g. [27,73,91]), or to create several process models from a single event log $\mathcal{W}$, each representing a subset of the recorded process behaviour (e.g. [24, 25, 137]).

Classification : for example, given k clusters, to which cluster does a new observation belong? From a process mining perspective, given a process model $\mathcal{M}$, was a new observed process trace x' generated by $\mathcal{M}$?

Decision trees can be used for classification, and also may be applied to understanding decision points in a process model [128]

Măruşter [108] made an early study of process mining from a machine learning viewpoint.

Machine learning techniques were used to cluster data in an event log in order to use it to create process models, and rules, metrics and algorithms were proposed for the construction of process models from sequences of activities. Herbst [77, 79] also takes a machine learning view, using Hidden Markov Models to infer workflow models from data.

Machine learning tasks can be classified as supervised or unsupervised, and active or passive. Under supervised learning, the learning machine receives feedback from training data. For example, learning a classifier C, the learning machine will be told whether it has classified an example x_i correctly, and can adjust its behaviour based on this feedback. Unsupervised learners, such as clustering algorithms, must infer the structure of the data without feedback. Related to this, active learners are able to ask questions, for example of an oracle, or select what data to use, whereas a passive learner simply uses the data it is given. Process mining is typically unsupervised and passive, making it difficult. One approach to alleviate this [67] artificially generates negative examples, which enables a classification algorithm to be used to construct a Petri net model.

Learning Theory

Process mining has been compared with the inference of grammars from example strings (e.g. [39, 122, 155]), the major difference being that grammars typically do not allow for concurrency. Nevertheless, the learning theory for grammar inference problems has been well studied and can be applied to the process mining field.

Machine learning theory is the study of questions such as the capability of machine learning algorithms, under what circumstances learning is possible, and what conditions are necessary to ensure an algorithm's success [104, Chapter 7].

Gold [69] introduced the paradigm of *Language Identification in the Limit*. A sequence of examples x from a grammar G is presented to a learner $\mathcal{L}$. After each example is presented, $\mathcal{L}$ attempts to reconstruct G. If $\mathcal{L}$ eventually generates a grammar G' equivalent to G, and G' does not change with further examples, then $\mathcal{L}$ is said to learn G in the limit. It was proven that only limited classes of languages are learnable under this criterion, es-

pecially when only positive examples (strings supported by G) are presented. Carrasco and Oncina [33] built on work by Angluin [13] to show that statistical regularity in the distribution of examples in G can compensate for the lack of negative examples, allowing Stochastic Regular Languages (representable by PDFA) to be learned. A heuristic algorithm is presented (see also [143]).

Valiant [75, 147] introduced the *Probably Approximately Correct (PAC)* paradigm for learning. Examples x generated randomly from grammar G (e.g. according to an underlying distribution P_G) are presented to the learner $\mathcal{L}$. G is PAC-learnable by $\mathcal{L}$ if with probability $(1 - \delta)$ it outputs grammar G', such that $d(G, G') < \epsilon$ for some measure of distance $d(\cdot, \cdot)$ between grammars, $0 < \delta, \epsilon \ll 1$, in time polynomial in $1/\delta$, $1/\epsilon$, and some measure of the complexity of G. This learning paradigm has been applied to learning PDFA, e.g. [180]. Clark and Thollard [36] discuss PAC-learnability of probabilistic automata and prove that PDFA are PAC-learnable given certain assumptions on the length of strings and distinguishability between states.

Other theories have been proposed, such as learning with queries to an oracle, mistake-bound learning and weighted majority. However in this thesis we consider the behaviour of process mining algorithms under the PAC framework due to the parallels between the stochastic regular grammars with which it was introduced, and our view of business processes. We discuss this next.

A Machine Learning View of Process Mining

A process discovery algorithm is essentially a learning machine, whose task is to model the control flow of a business process, using traces of the execution of the process, recorded in an event log $\mathcal{W}$, which is a multi-set over traces. Each trace represents a single run through the process from start to end. Traces can be encoded as strings $x \in \Sigma^+$, where Σ is an alphabet of symbols representing activities.

We assume that an unknown probability distribution $\mathcal{D}$ over traces (from Σ^+) is responsible for generating the traces in the log $\mathcal{W}$. Although various factors affect what

activities take place, such as business needs or user preferences, different traces in fact occur with specific probabilities, and thus it can be argued that the underlying process is inherently stochastic. From the machine learning point of view, the primary task of the process mining algorithm is to construct a model $\mathcal{M}$ of $\mathcal{D}$ from a finite sample of traces (event log $\mathcal{W}$).

The log file $\mathcal{W}$ will contain only a finite number of process traces, and therefore is a stochastic sample drawn *i.i.d.* (independently and identically distributed) from the unknown distribution $\mathcal{D}$ (the 'ground truth'). In other words, each trace occurs with probability according to the same distribution $\mathcal{D}$, and one trace occurring does not change the probability of others. Since the log is of finite size, we expect the frequency of traces in the log to vary from their probabilities under $\mathcal{D}$.

The challenge for the learning machine (process discovery algorithm) is to use this finite sample to construct a model $\mathcal{M}$ of $\mathcal{D}$ which does not simply represent the data in the finite log, but is as 'close' as possible to the true generating source $\mathcal{D}$, i.e. generalises well. This raises questions such as: *How much data is needed to do this with certain (given) confidence and precision*[1]*? How to quantify the learning machine's performance?* Since both $\mathcal{D}$ and $\mathcal{M}$ are distributions, it is natural to assess the learning machine's performance by quantifying how 'close' $\mathcal{M}$ is to $\mathcal{D}$, for which there are various measures. This allows direct comparison of the 'reality' represented by the models, rather than similarity/dissimilarity of syntactic representations of $\mathcal{M}$ and $\mathcal{D}$ in the modelling language in use, which seems to be a common theme [12, 16, 46, 70] (Section 3.3).

Machine learning theory is concerned with the convergence properties of machine learning algorithms, in terms of the circumstances in which they can be expected to converge to the ground truth, and the amount of data needed. While different process discovery algorithms have different strengths and weaknesses, they can be compared under this unifying framework, i.e. in terms of their convergence properties within the restrictions within which they operate. From the ground truth and an understanding of the behaviour

[1]This corresponds to the PAC framework.

of an algorithm, one can predict, and experimentally verify, whether the mined model will converge to the ground truth model, and how fast it will do so.

While in real applications the process discovery algorithm will not have access to the ground truth distribution governing trace generation, it is standard practice in machine learning [14, 104] to study learning algorithms by imposing a certain class of ground truth distributions and then to verify empirically and/or theoretically how fast and how well the ground truth can be 'learnt' by the algorithm from finite samples. In this framework, the algorithm does not know the ground truth, but because we have access to it, the success of the learning algorithm as more samples become available can be measured.

4.2 Framework for the Analysis of Process Mining Algorithms

In this section we outline a framework within which to analyse process mining algorithms with regard to their probabilistic behaviour, process sub-structures, and number of traces; in the context of their ability to discover a probability distribution over traces, which converges to a 'ground truth'. To analyse algorithms, we assume that we have access to this ground truth and know probabilities of all strings (sequences of activities).

The steps below describe the approach taken here to analyse and experimentally validate process mining algorithms.

Step 1. Analyse the algorithm to develop formulae for the probability of discovery of all important process sub-structures (e.g. splits and joins, or parallel action flows), based on these string probabilities, agnostic of whether these sub-structures are 'correct', i.e. assuming nothing about the underlying model, save that it is acyclic, and traces are generated according to an unknown probability distribution.

Step 2. Extend to aggregate the sub-structure results (joining sub-structures from the previous step into the full model) to enable calculation of overall discovery probability of arbitrary models.

Step 3. Analyse the algorithm's characteristics, such as rate of convergence, issues affecting convergence, possible relation to other algorithms, etc.

Theoretical analysis will be complemented by empirical investigations as follows:

1. Design 'ground truth' test models with varying topological and probability structures. Depending on the complexity of the designed models, probabilities of strings and sub-strings $x \in \Sigma^+$ may be read from the model, or estimated by counting the number of times x occurs in traces in a 'large' log of n traces randomly simulated from the designed model, e.g. $\pi(x) \approx \frac{N(x)}{n}$.
2. From the test models generate multiple sample sets of event logs of various sizes, to test for convergence.
3. Run the process mining algorithms under investigation on such data, converting mining results to PDFA as necessary (Sections 2.3.1 and 2.3), and compare distributions of traces represented by these automata with the 'ground truth'[1].

In the following subsections we introduce the important process sub-structures, then in Chapter 5 we apply the framework to an analysis of the Alpha algorithm [156].

4.2.1 Process Sub-Structures

Business processes are composed of sub-structures [133, 154]. We consider only acyclic structures in this thesis. A few basic structures are sufficient, although more complex patterns exist [154]. The sub-structures in our example process are highlighted in Figures 4.1 (Petri net) and 4.2 (equivalent PDFA).

In the next sections we define process sub-structures in terms of the restrictions which they imply on the set $\mathcal{T}$ of valid process traces starting with i and ending with o, and on the probability distribution over traces P_M.

[1]Where algorithms produce non-probabilistic models, heuristic methods can be used to allocate uniform or maximum likelihood probabilities to transitions.

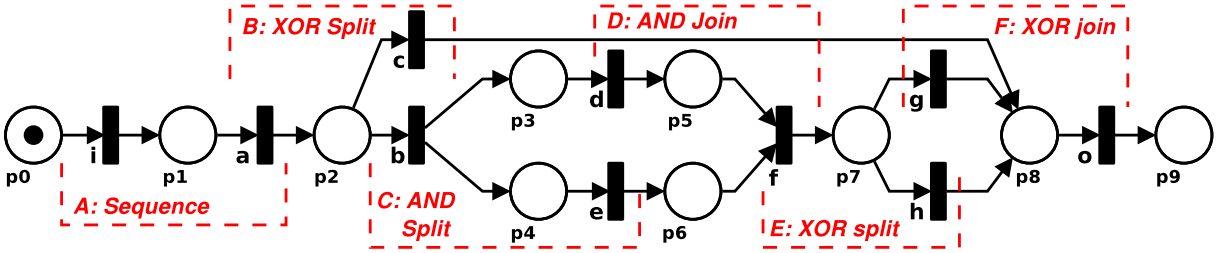


Figure 4.1: Example of Process Sub-Structures in Petri Net N_0.

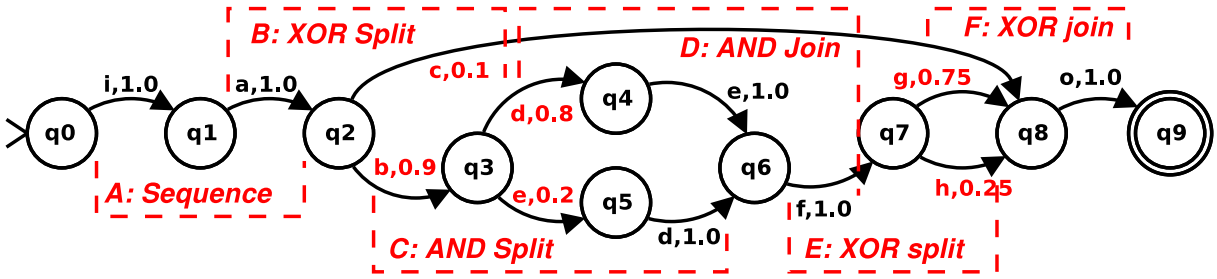


Figure 4.2: Example of Process Sub-Structures in PDFA A_0.

Sequences

If activities a and b form a sequence, $a \rightarrow b$, then if a occurs, it is immediately followed by activity b, and no other, in the model. Activities b and a cannot occur in the reverse order. Figures 4.3, 4.4 show Petri net and PDFA fragments respectively, depicting the ab sequence. Structure A in the example process (Figures 4.1, 4.2) represents sequence ia.

In the event log, other parallel activities may 'interfere', so $\pi(b|a) \leq 1$. The following will hold: if a occurs in a trace, b will occur before the end of the trace, i.e.:

$$\text{if } uav \in \mathcal{T}, \text{ then } v = wbq,$$
$$\text{where } a, b \in \Sigma, \text{ and } u, w, q \in \{\Sigma \setminus \{a, b\}\}^*.$$

The sequence imposes the following restrictions on the distribution over traces:

- $\pi(ab) \geq 0$. Although according to the model, b always follows a, nevertheless it may be that ab does not occur in any traces in $\mathcal{W}$ (due to occurrence of other parallel activities). This is a problem for many algorithms such as Alpha [156] and Heuristics Miner [194] which derive a model using local relations between activities in $\mathcal{W}$.

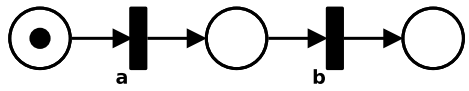


Figure 4.3: Sequence: Petri Net Fragment.

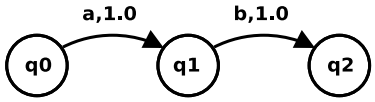


Figure 4.4: Sequence: PDFA Fragment.

- $\forall u \in \{\Sigma \setminus \{a,b\}\}^*$, $\exists w \in \{\Sigma \setminus \{a,b\}\}^*$ s.t. $\pi(wb|ua) = 1$. If a trace contains a, then b must occur before the end of the trace.
- $\pi(bua) = 0, u \in \{\Sigma \setminus \{a,b\}\}^*$. If b can occur before a, a and b do not form a sequence.

In the running example process (Figure 4.1), no other activities occur in parallel with ia, so we see every trace in the example log (Table 3.2) begins with ia and none contain ai, i.e. suggesting that for this model, $\pi(ia) = 1$ and $\pi(ai) = 0$.

Exclusive-OR Split

An m-way XOR split, $a \rightarrow (b_1 \# \ldots \# b_m)$ (Petri net Figure 4.5, PDFA Figure 4.6), occurs where there is a choice between m mutually exclusive paths through the model after activity a, each path starting with an activity $t \in \{b_i | 1 \leq i \leq m\}$. If a occurs in a trace, then exactly one $t \in \{b_i | 1 \leq i \leq m\}$ will occur in the rest of the trace:

$$\text{if } uav \in \mathcal{T}, \text{ then } \exists i : 1 \leq i \leq m, \text{ such that } v = wb_iq,$$
$$\text{where } a, b_i \in \Sigma, \text{ and } u, w, q \in \{\Sigma \setminus \{a, b_1, \ldots, b_m\}\}^*.$$

The XOR split places the following restrictions on the PDF:

- $\pi(ab_i) \geq 0, \forall\ 1 \leq i \leq m$. Although the model allows any b_i to directly follow a, as with sequences, the probability may be zero for some such sequences occurring in the log; a problem for many algorithms.
- $\forall 1 \leq i \leq m$, $\forall u \in \{\Sigma \setminus \{a, b_1, \ldots, b_m\}\}^*$, $\exists w \in \{\Sigma \setminus \{a, b, \ldots, b_m\}\}^*$ s.t. $\pi(wb_i|ua) > 0$. Given that a occurs in the trace, it must be possible for any b_i to appear in the remainder of the trace.
- $\pi(b_iua) = 0, \forall\ 1 \leq i \leq m, u \in \{\Sigma \setminus \{a, b_i\}\}^*$. As for sequences, activities before and

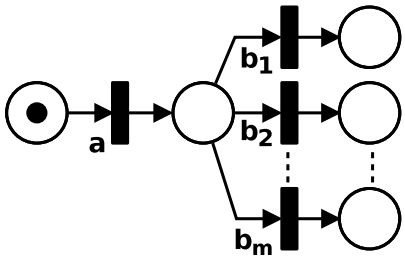


Figure 4.5: XOR Split: Petri Net.

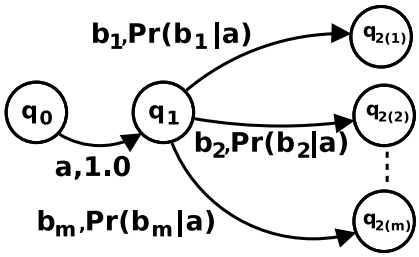


Figure 4.6: XOR Split: PDFA.

after the split cannot occur in the reverse order.

- $\pi(b_i u b_j) = 0, \forall\ 1 \leq i < j \leq m, u \in \{\Sigma \setminus \{a, b_i, b_j\}\}^*$. If the trace contains multiple 'post-split' activities b_i, b_j, etc. then these are not exclusive.

Structures B and E in the example process represent two-way splits, from activity a to either b or c, and from activity f to g or h respectively.

In Table 3.2, each trace contains a, immediately followed by either b or c (no other activities can occur in parallel). In the traces that contain f, it is always followed (immediately) by g or h. From this small sample, b and c do not occur together in traces, nor do g and h. Thus as described above, $\pi(ab) > 0$, $\pi(ac) > 0$, $\pi(fg) > 0$, $\pi(fh) > 0$ and we could estimate that $\pi(ba) = \pi(ca) = \pi(bc) = \pi(cb) = \pi(gf) = \pi(hf) = \pi(gh) = \pi(hg) = 0$.

Exclusive-OR Join

An m-way XOR join, $(b_1 \# \ldots \# b_m) \rightarrow c$, occurs where m mutually exclusive paths rejoin before activity c. The final activity in each path prior to c is a activity $t \in \{b_i | 1 \leq i \leq m\}$. If c occurs in a trace, then exactly one activity $t \in \{b_i | 1 \leq i \leq m\}$ will be in the trace before c:

$$\text{if } ucv \in \mathcal{T}, \text{ then } \exists i : 1 \leq i \leq m, \text{ such that } u = wb_iq,$$
$$\text{where } c, b_i \in \Sigma, \text{ and } v, w, q \in \{\Sigma \setminus \{c, b_1, \ldots, b_m\}\}^*.$$

Similar restrictions are placed on the PDF to those imposed by the split:

- $\pi(b_i c) \geq 0, \forall\ 1 \leq i \leq m$. Although the model allows any b_i to directly precede c,

again probabilities may be zero for seeing some of these sequences in the log.

- $\pi(c|wb_iq) = 1, \forall\ 1 \leq i \leq m, w, q \in \{\Sigma \setminus \{c, b_1, \ldots, b_m\}\}^*$. If any b_i occurs in the trace, then c must appear in the remainder of the trace.
- $\pi(cub_i) = 0, \forall\ 1 \leq i \leq m, u \in \{\Sigma \setminus \{a, b_i\}\}^*$. As for sequences, activities before and after the split cannot occur in the reverse order.
- $\pi(b_iub_j) = 0, \forall\ 1 \leq i < j \leq m, u \in \{\Sigma \setminus \{a, b_i, b_j\}\}^*$. If the trace contains multiple 'pre-split' activities b_i, b_j, etc. then these are not exclusive.

Structure F in the example process represents a three-way join from activities c, g and h to o. Referring to the example traces in Table 3.2, c, g and h are always followed by o, with none of the disallowed sequences of activities such as og or gc.

Parallel Split

An m-way parallel (AND) split, $a \rightarrow (b_1 \parallel \ldots \parallel b_m)$ (Petri net Figure 4.7, PDFA Figure 4.8), occurs where m paths through the model proceed in parallel, following activity a, each path starting with a activity $t \in \{b_i | 1 \leq i \leq m\}$. If each path contains only a single activity b_i, and there are no restrictions on the order of the activities, and no other parallel parts of the model, then the next m activities in the trace will be $b_1, b_2, \ldots, b_m$, in one of $m!$ permutations. Otherwise, there will be more possibilities for the trace following a. In reality, it is likely that only a subset of the possible orderings will be highly probable. If a occurs in a trace, then the remainder of the trace following a will contain each $t \in \{b_i | 1 \leq i \leq m\}$:

$$\text{if } uav \in \mathcal{T}, \text{ then } \forall i : 1 \leq i \leq m, (\exists w, q \in \{\Sigma \setminus \{a, b_i\}\}^*, \text{ such that } v = wb_iq),$$
$$\text{where } a, b_i \in \Sigma, u \in \{\Sigma \setminus \{a, b_1, \ldots, b_m\}\}^*.$$

The parallel split places similar restrictions on the PDF to those imposed by XOR splits:

- $\pi(ab_i) \geq 0, \forall\ 1 \leq i \leq m$. As for XOR, it must be possible for any b_i to directly follow a, but some probabilities of seeing the sequences in the log, may be zero.

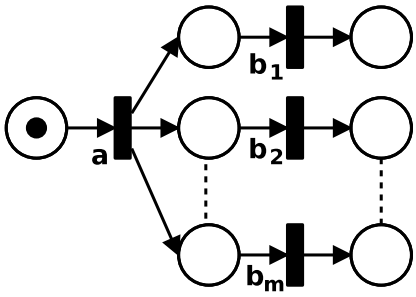


Figure 4.7: Parallel (AND) Split: Petri Net.

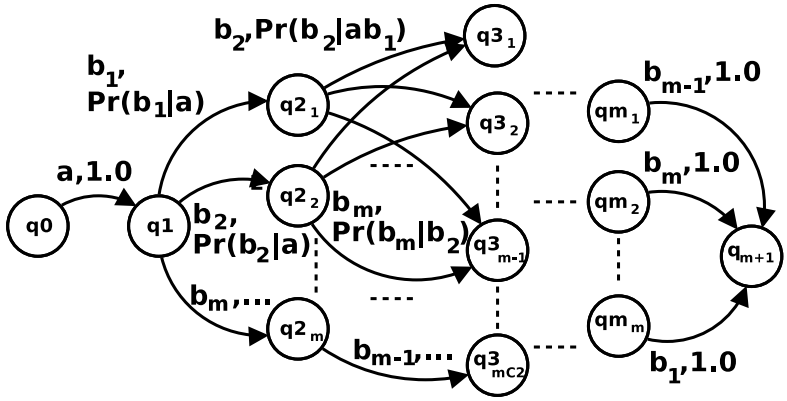


Figure 4.8: Parallel (AND) Split: PDFA.

- $\forall u \in \{\Sigma \setminus \{a, b_1, \ldots, b_m\}\}^*$, $\forall 1 \leq i \leq m$, $\exists w \in \{\Sigma \setminus \{a, b_i\}\}^*$ s.t. $\pi(wb_i|ua) = 1$. Given that a occurs in the trace, each b_i must appear in the remainder of the trace (in some order).
- $\pi(b_i ua) = 0, \forall\ 1 \leq i \leq m, u \in \{\Sigma \setminus \{a, b_i\}\}^*$. As for sequences, activities before and after the split cannot occur in the reverse order.
- $\pi(b_i u b_j) > 0, \forall\ 1 \leq i < j \leq m, u \in \{\Sigma \setminus \{a, b_i, b_j\}\}^*$. Conversely to XOR splits, it must be possible for multiple 'post-split' activities b_i, b_j, to occur in a trace, else they are not in parallel.
- $\forall u \in \{\Sigma \setminus \{a, b_1, \ldots, b_m\}\}^*$, $\forall 1 \leq i < j \leq m$, $\exists v, w \in \{\Sigma \setminus \{a, b_i, b_j\}\}^*$ s.t. $\pi(vb_i wb_j|ua) = 1$. This extends the previous restriction: if a occurs, then the remainder of the trace must contain all b_i.

A PDFA fragment to depict a parallel split is visually more complex than its Petri net equivalent, as all possible activity sequences are shown explicitly (Figure 4.8). After the first parallel activity there are $\binom{m}{1}$ states, $\binom{m}{2}$ after the second, to $\binom{m}{m-1}$ states before the last parallel activity. Structure C in the example process represents a two-way parallel split from activity b to d and e in parallel.

In Table 3.2, both *de* and *ed* can occur in traces. If either d or e occurs, they both do. Since in this simple model there are no other parts of the model in parallel, no activities occur between d and e. Therefore from this small sample we could estimate

$\pi(d|e) = \pi(e|d) = 1$. Neither of the 'disallowed' strings db or eb occur.

Parallel Join

An m-way parallel (AND) join, $(b_1 \parallel \ldots \parallel b_m) \rightarrow c$, occurs where m parallel paths rejoin (synchronise) before a activity c. The final activity in each path is one of $b_1, b_2, \ldots, b_m$. If c occurs in a trace, then the trace up to c will contain each $t \in \{b_i | 1 \leq i \leq m\}$:

$$\text{if } ucv \in \mathcal{T}, \text{ then } \forall i : 1 \leq i \leq m, (\exists w, q \in \{\Sigma \setminus \{c, b_i\}\}^*, \text{ such that } u = wb_iq),$$
$$\text{where } c, b_i \in \Sigma, v \in \{\Sigma \setminus \{c, b_1, \ldots, b_m\}\}^*.$$

The parallel split places similar restrictions on the PDF:

- $\pi(b_ic) \geq 0, \forall\ 1 \leq i \leq m$. As for XOR, it must be possible for any b_i to directly follow a.
- $\pi(c|wb_iq) = 1, \forall\ 1 \leq i \leq m, w, q \in \{\Sigma \setminus \{c, b_i\}\}^*$. If any b_i occurs in the trace, then c must appear in the remainder of the trace.
- $\pi(cub_i) = 0, \forall\ 1 \leq i \leq m, u \in \{\Sigma \setminus \{c, b_i\}\}^*$. Activities before and after the split cannot occur in the reverse order.
- $\pi(b_iub_j) > 0, \forall\ 1 \leq i < j \leq m, u \in \{\Sigma \setminus \{a, b_i, b_j\}\}^*$. Conversely to XOR joins, it must be possible for multiple 'pre-split' activities b_i, b_j, to occur in a trace, else they are not in parallel.

Structure D in the example process represents a two-way parallel join from parallel activities d and e to f. Table 3.2 shows that as for the split, d and e occur together, and are followed by f. Activity f does not precede d or e.

Non-Exclusive OR Splits and Joins

These occur where one or many of several paths may be taken. They can be modelled as combinations of XOR and parallel structures, and place corresponding restrictions on the distribution over traces. In this thesis we do not consider such splits and joins further.

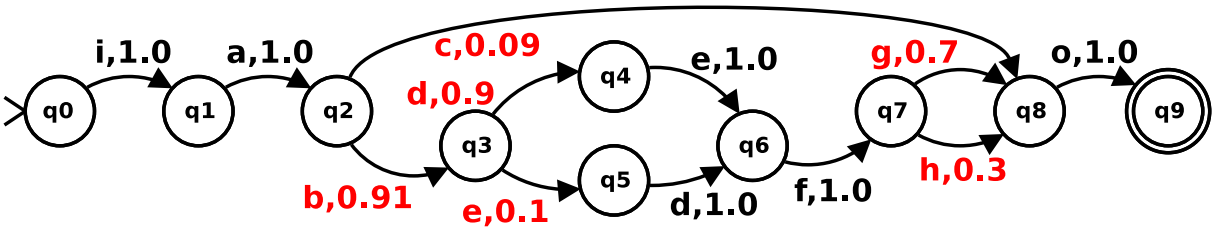


Figure 4.9: PDFA A_1 differing from A_0 (Figure 2.5) in Probabilities only.

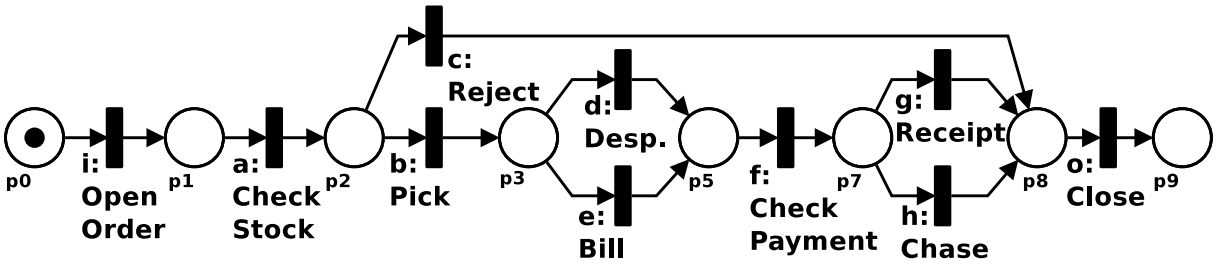


Figure 4.10: Petri Net N_1 structurally different from N_0 (Figure 3.1).

4.3 Discussion of Measures for Assessment of Process Mining Results

In this section we use the running example to compare distances between distributions, with existing Petri net based process mining metrics, and show that distances give a clearer view of how different two process models are. This is confirmed by the experimentation in the next chapter (Sections 5.4.1 and 5.4.2).

Let PDFA A_0 (Figure 2.5) describe the ground truth distribution over process traces for a simple process. Figure 4.9 shows PDFA A_1 produced by a hypothetical process mining algorithm $\mathcal{L}_1$, mining from a particular log $\mathcal{W}_1$, a finite sample from the ground truth distribution. The trace frequencies in $\mathcal{W}_1$ vary from the ground truth probabilities, preventing $\mathcal{L}_1$ from creating PDFA A_1 with the exact ground truth probabilities.

Petri net N_0 (Figure 3.1) models the same process without probability information, in that it supports the same set of traces as the ground truth. The Petri net can be compared with the ground truth by converting to a PDFA, by labelling its reachability graph (Figure 2.5) with probabilities (Sections 2.3.1 and 2.3).

Another algorithm $\mathcal{L}_2$ might mine a Petri net directly, producing net N_1 (Figure 4.10). This algorithm has failed to discover the parallelism, instead using an XOR split/join.

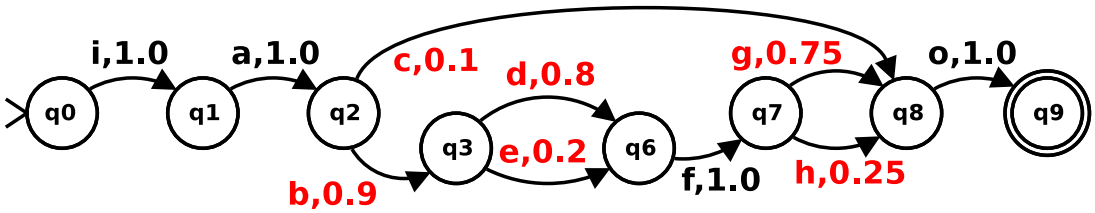


Figure 4.11: PDFA A_2 corresponding to Petri Net N_1 (Figure 4.10).

Models	$1 - d_{Bhat}$	$1 - d_2/\sqrt{2}$	$1 - d_{JSD}/2$	**Fitness** f
$A_0 : A_1$	0.897	0.926	0.985	1.0
$A_0 : A_2$	0.051	0.413	0.1	0.893

Table 4.1: Illustration of Distances between Process Models.

This net, and its corresponding PDFA A_2 (Figure 4.11), are structurally different from the ground truth, therefore supporting a different set of traces. This is a serious problem, as this model does not allow for both despatch of the product and billing.

Table 4.1 shows the distances between these PDFA and the ground truth, using the distance measures described (scaled and subtracted from 1 to allow comparison with Fitness f). Models A_0, A_1 are measured as quite similar, but A_0, A_2 as almost 100% different. Although structurally 'similar', they support fundamentally different behaviour since the split/join type has been changed. What is more, this part of the model accounts for 90% of the probable traces. Conversely, Fitness f measures A_2 as relatively well fitting. Although it takes account of the frequency of non-fitting traces, it penalises traces only (approximately) at the level of the non-fitting events. Thus, effectively, only event d or e is penalised in a non-fitting trace, while i, a, b, c, f, g, h, o are not. This leads to a misleading picture of the correctness of this model.

This can be seen further in the graph in Figure 4.12. Here we varied the probability of the part of the model containing the parallel sub-structure. The graph shows the closeness of the mined model to the ground truth, for the various metrics, as the probability of the parallel part of the model varies from very low, to very high. Figure 4.12 suggests the distance metrics to be more analysable (Section 3.3.1 and [124]) than Fitness (f), measuring the mined model to be almost optimal where the parallel sub-structure is

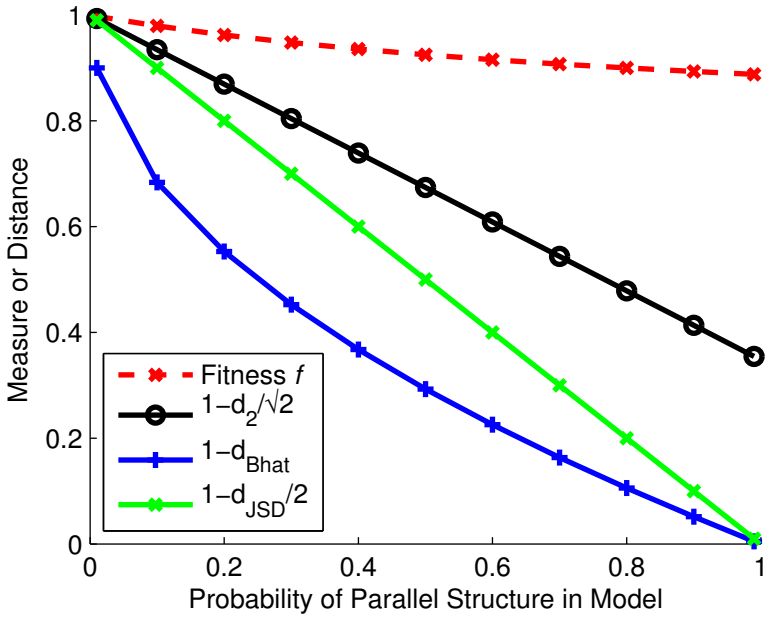


Figure 4.12: Comparison of Metrics: varying Probability of Parallel Sub-Structure.

unlikely to be involved (where the error in the model does not affect many traces), reducing to zero as traces involving the parallel sub-structure form the majority of the behaviour.

4.4 Chapter Summary

This chapter introduced the probabilistic framework for analysing process mining algorithms, which is the core theoretical contribution of the thesis. We described representing business processes as distributions over strings of symbols, and representing these distributions using probabilistic automata. We motivated this view by describing a machine learning view of process mining as the learning of probability distributions over strings of symbols, from samples from the underlying distributions. Finally, we described basic process sub-structures in terms of the restrictions which they imply on such distributions.

In the next two chapters we apply this framework to the theoretical analysis of two well-known process mining algorithms. We show that this provides insight into the learning behaviour of these algorithms, and provides a basis for practical applications of our method in subsequent chapters.

Part II

Applications of the Framework

CHAPTER 5

CASE STUDY: ANALYSIS OF THE ALPHA ALGORITHM

In this chapter we use the probabilistic framework described in the previous chapter, to analyse the well-known Alpha process mining algorithm [148,156]. We apply the analysis to our simple running example, and to a more complex model illustrative of larger systems. The experimentation shows that our method gives insights into the behaviour of the Alpha algorithm when mining these models.

Alpha was an early process mining algorithm, developed to mine concurrent processes and formally proven to correctly mine processes representable by Structured WF-Nets (SWF-Nets), from noise-free logs[1]. These limitations mean that it is not regarded as a practical mining algorithm in real-world situations. However, it is in fact often used, due to its simplicity, which also makes it appropriate for a first analysis under our framework. It is also used as the basis for several other algorithms (e.g. [48, 85, 92, 117, 195]).

Much of the material in this chapter was originally presented in [184].

5.1 A Probabilistic Analysis of the Alpha Algorithm

The Alpha algorithm was described in Chapter 3. The four Alpha relations $\rightarrow$, $\rightarrow^{-1}$, $\#$ and $\|$, on a pair of activities a, b partition the set of all logs of n traces, as illustrated in

[1] In this chapter we informally understand 'noise-free' event logs as being recorded without error by an underlying process which is followed without error. We propose a formal definition in Chapter 8.

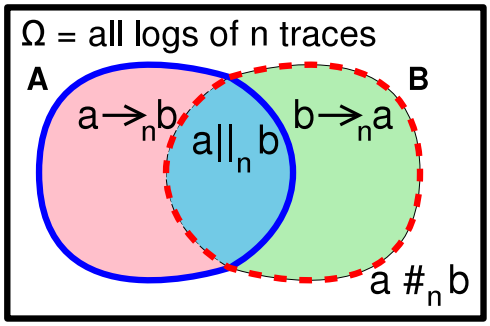


Figure 5.1: The Alpha Relations on a Pair of Activities Partition the possible Logs of n traces.

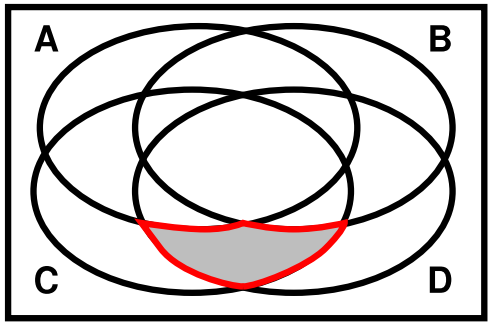


Figure 5.2: Illustration of $(C \cap D) \setminus (A \cup B)$ for Proposition 5.

Figure 5.1. In this chapter we write the relations as $a >_n b$, etc., to indicate discovery within n traces. Consider event space Ω, the set of all logs of n traces. Let A be the set of these logs that include at least one trace containing sub-string ab ($a >_n b$), and B those with at least one trace with ba ($b >_n a$). Then

- $A \setminus B$ is the set of logs that cause Alpha to infer the causal relation $a \rightarrow_n b$,
- $B \setminus A$ those for which Alpha infers $b \rightarrow_n a$,
- $A \cap B$ those for which Alpha infers $a \parallel_n b$, and
- $\neg(A \cup B)$ those for which Alpha infers $a \#_n b$.

In the following section we apply the steps described in Chapter 4 Section 4.2 to the Alpha algorithm.

5.2 Step 1: Probability Formulae for Basic Process Sub-Structures

Under the assumptions in Chapter 4, we have access to the ground truth and knowledge of probabilities of all sequences of activities, we next give formulae for the probability of discovery of the Alpha relations and process sub-structures.

5.2.1 Activity Ordering Relations

These are the basic relations between activities in the log which Alpha uses to construct a Petri net model[1]. The following Propositions give the probability of Alpha inferring these relations between two activities a and b, from a log of n traces, based on the sub-string probabilities described in Section 4.1. We give the proofs in Appendix A.1.

We use $P_{\alpha,n}(a >_n b)$ to denote the probability that Alpha infers the relation $a > b$ over n traces, and similarly for the other Alpha relations.

Proposition 1. *The probability that Alpha infers* $a >_n b$ *is*

$$P_{\alpha,n}(a >_n b) = 1 - \big(1 - \pi(ab)\big)^n.$$

Proposition 2. *The probability that Alpha infers* $a \#_n b$ *is*

$$P_{\alpha,n}(a \#_n b) = \big(1 - \pi(ab) - \pi(ba)\big)^n.$$

Proposition 3. *The probability that Alpha infers* $a \to_n b$ *is*

$$P_{\alpha,n}(a \to_n b) = \big(1 - \pi(ba)\big)^n - \big(1 - \pi(ab) - \pi(ba)\big)^n. \tag{5.1}$$

Proposition 4. *The probability that Alpha infers* $a \parallel_n b$ *is*

$$P_{\alpha,n}(a \parallel_n b) = 1 - \big(1 - \pi(ab)\big)^n - \big(1 - \pi(ba)\big)^n + \big(1 - \pi(ab) - \pi(ba)\big)^n.$$

5.2.2 Sequences

Discovery of a basic sequence of two activities a and b (Petri net Figure 4.3, PDFA Figure 4.4) simply requires discovery of $a \to_n b$.

[1] Alpha$^+$, implemented in the ProM framework [170] as Alpha, modifies these to allow for short loops.

5.2.3 Splits and Joins

Alpha uses the relations $a \rightarrow_n b$, $a \#_n b$ and $a \parallel_n b$ to identify Petri net places, which determine the types of splits and joins (XOR or AND). Since the events of the discovery of these relations between several activities arise from Alpha's interpretation of a log of n traces, they are not independent: any, all or no relations may be discovered. Thus $P_{\alpha,n}\big((a \rightarrow_n b) \wedge (a \rightarrow_n c)\big) \leq P_{\alpha,n}(a \rightarrow_n b) \times P_{\alpha,n}(a \rightarrow_n c)$. Therefore for exact probabilities for discovery of splits and joins, the basic sub-string probabilities (probabilities of activity pairs which *must*/*must not* be seen in the log) must be used.

5.2.4 Exclusive Choice: XOR Split

To discover an m-way XOR split from a to $b_1, b_2, \ldots, b_m$ (Petri net Figure 4.5, PDFA Figure 4.6), denoted $a \rightarrow_n (b_1 \# \ldots \# b_m)$: Alpha must infer the relations $a \rightarrow_n b_1$, $a \rightarrow_n b_2$, $\ldots$, $a \rightarrow_n b_m$, $b_1 \#_n b_2$, $b_1 \#_n b_3$, $\ldots$, $b_{m-1} \#_n b_m$ [156, Defn 4.3 step 4]. So over a log of n traces, Alpha must:

- see at least one of each of m sub-strings $ab_1, ab_2, \ldots, ab_m$ representing pairs of activities; and
- not see any of the m 'reverse' pairs $b_1a, b_2a, \ldots, b_ma$, or any of ${}_mP_2$ pairs of 'post-split' activities: b_1b_2, b_2b_1, $\ldots$, $b_{m-1}b_m$, b_mb_{m-1} $\big($where ${}_mP_2 \triangleq \frac{m!}{(m-2)!}\big)$.

Let $N = \{N_i = (t_i, t'_i) | 1 \leq i \leq (m + {}_mP_2), t_i \neq t'_i\}$ be the set of activity pairs which *must not* be seen in the log, and $Y = \{Y_i = (t_i, t'_i) | 1 \leq i \leq m, t_i \neq t'_i\}$ be the set of activity pairs which *must* be seen in the log.

We define $S_n(X) \rightarrow [0, 1]$, where $X = \{X_i = (t_i, t'_i) | 1 \leq i \leq |X|\}$ is the probability of *not* seeing any of the $|X|$ activity pairs $(t_i, t'_i) \in X$ in n traces, and $\pi(X_i) = \pi(t_i t'_i)$.

Proposition 5. *The probability that Alpha infers an XOR split is*

$$P_{\alpha,n}\big(a \rightarrow_n (b_1 \# \ldots \# b_m)\big) = S_n(N) - \sum_{1\leq i\leq m} S_n\big(N \cup \{Y_i\}\big) + \sum_{1\leq i<j\leq m} S_n\big(N \cup \{Y_i, Y_j\}\big) - \ldots + (-1)^m S_n(N \cup Y), \tag{5.2}$$

where

$$S_n(X) = \Bigg(1 - \sum_{1\leq i\leq |X|} \pi(X_i) + \sum_{1\leq i<j\leq |X|} \pi(X_i \wedge X_j) - \ldots + (-1)^{|X|}\pi(X_1 \wedge X_2 \wedge \ldots \wedge X_{|X|})\Bigg)^n. \tag{5.3}$$

Given knowledge about the underlying model, many of the terms may be zero, significantly simplifying the formulae. Nevertheless, they can become cumbersome to work with, requiring knowledge of many probabilities. Nor do they relate intuitively to the working of the algorithm. However, these formulae can be effectively simplified without loss of accuracy to give formulae which intuitively follow from the working of the Alpha algorithm, and are simpler to calculate. Theorem 1 illustrates for Proposition 5, the discovery of an XOR split.

Theorem 1. *The probability of discovery of an XOR split may be upper bounded by assuming independence between discovery of Alpha relations over n traces. The probability is over-stated but error rate decreases exponentially with increasing n:*

$$P_{\alpha,n}\big(a \rightarrow_n (b_1 \# \ldots \# b_m)\big) \leq \prod_{1\leq i\leq m} P_{\alpha,n}(a \rightarrow_n b_i) \times \prod_{1\leq i<j\leq m} P_{\alpha,n}(b_i \#_n b_j). \tag{5.4}$$

The proof is given in Appendix A.1. In what follows we use the upper bound in Theorem 1 to derive formulae for discovery of XOR joins, and analogous results (not presented here) for AND splits and joins.

5.2.5 Exclusive Choice: XOR Join

This is similar to the XOR split. To discover a m-way join between exclusive paths from $b_1, b_2, \ldots, b_m$ to c, denoted $(b_1 \#_n \ldots \#_n b_m) \rightarrow_n c$: Alpha must infer the relations $b_1 \rightarrow_n c$, $b_2 \rightarrow_n c$, $\ldots$, $b_m \rightarrow_n c$, $b_1 \#_n b_2$, $b_1 \#_n b_3$, $\ldots$, $b_{m-1} \#_n b_m$, so over a log of n traces, Alpha must:

- see at least one of each of m sub-strings $b_1c, b_2c, \ldots, b_mc$ representing pairs of activities; and
- not see any of the m 'reverse' pairs $cb_1, cb_2, \ldots, cb_m$, or any of ${}_mP_2$ pairs of 'pre-split' activities: $b_1b_2, b_2b_1, \ldots, b_{m-1}b_m, b_mb_{m-1}$.

N, Y, $S_n(X)$ are defined as for the XOR split, and $P_{\alpha,n}\big((b_1 \# \ldots \# b_m) \rightarrow_n c\big)$ as for equation (5.2), for the appropriate pairs of activities:

The discovery of an m-way XOR join can be bounded in a similar way:

$$P_{\alpha,n}\big((b_1 \# \ldots \# b_m) \rightarrow_n c\big) \leq \prod_{1 \leq i \leq m} P_{\alpha,n}(b_i \rightarrow_n c) \times \prod_{1 \leq i < j \leq m} P_{\alpha,n}(b_i \#_n b_j).$$

5.2.6 Parallelism: AND Split

The behaviour of the Alpha algorithm when mining an AND split (Petri net Figure 4.7, PDFA Figure 4.8) is similar to that when mining an XOR split, with more 'must see' and fewer 'must not see' sub-strings. To discover a m-way parallel split from a to $b_1, b_2, \ldots, b_m$, denoted $a \rightarrow_n (b_1 \parallel \ldots \parallel b_m)$: Alpha must infer the relations $a \rightarrow_n b_1$, $a \rightarrow_n b_2$, $\ldots$, $a \rightarrow_n b_m$, $b_1 \parallel_n b_2$, $b_1 \parallel_n b_3$, $\ldots$, $b_{m-1} \parallel_n b_m$ [156, Defn 4.3 step 4].

For the parallel split, now $N = \{N_i = (t, t') | 1 \leq i \leq m, t \neq t'\}$ is the set of activity pairs which *must not* be seen in the log, and $Y = \{Y_i = (t, t') | 1 \leq i \leq (m + {}_mP_2), t \neq t'\}$ the set of activity pairs which *must* be seen in the log. The probability of mining the parallel split is given by equation (5.5) with the modified sets N and Y, i.e..

Proposition 6. *The probability that Alpha infers a parallel (AND) split is*

$$P_{\alpha,n}\big(a \rightarrow_n (b_1 \parallel \ldots \parallel b_m)\big) = S_n(N) - \sum_{1 \le i \le m +_m P_2} S_n\big(N \cup \{Y_i\}\big) + \sum_{1 \le i < j \le m +_m P_2} S_n\big(N \cup \{Y_i, Y_j\}\big) - \ldots + (-1)^m S_n(N \cup Y), \quad (5.5)$$

where $S_n(X)$ is again defined as in Equation (5.3). It can be bounded similarly,

$$P_{\alpha,n}\big(a \rightarrow_n (b_1 \parallel \ldots \parallel b_m)\big) \le \prod_{1 \le i \le m} P_{\alpha,n}(a \rightarrow_n b_i) \times \prod_{1 \le i < j \le m} P_{\alpha,n}(b_i \parallel_n b_j).$$

5.2.7 Parallelism: AND Join

This is similar to the AND split. To discover a m-way parallel join from $b_1, b_2, \ldots, b_m$ to c, denoted $(b_1 \parallel_n \ldots \parallel_n b_m) \rightarrow_n c$: Alpha must infer the relations $b_1 \rightarrow_n c$, $b_2 \rightarrow_n c$, $\ldots$, $b_m \rightarrow_n c$, $b_1 \parallel_n b_2$, $b_1 \parallel_n b_3$, $\ldots$, $b_{m-1} \parallel_n b_m$.

N, Y, $S_n(X)$ are defined as for the AND split, and $P_{\alpha,n}\big((b_1 \parallel \ldots \parallel b_m) \rightarrow_n c\big)$ as for equation (5.5), for the appropriate pairs of activities.

Similarly the following bound applies:

$$P_{\alpha,n}\big((b_1 \parallel \ldots \parallel b_m) \rightarrow_n c\big) \le \prod_{1 \le i \le m} P_{\alpha,n}(b_i \rightarrow_n c) \times \prod_{1 \le i < j \le m} P_{\alpha,n}(b_i \parallel_n b_j).$$

5.3 Step 2: Aggregation of Sub-Structures to Full Model

Having dealt with sub-structures, we now need to derive probability $P_{\alpha,n}(\mathcal{M})$ of correctly mining the full process model $\mathcal{M}$ (e.g. Figure 4.1 with sub-structures labelled $A \ldots F$). For exact calculation the approach of Proposition 5 could be extended to consider the probabilities of all the sub-strings which Alpha must or must not see in the log to construct the Petri net correctly, as these probabilities are not independent (Section 5.2.3). This is

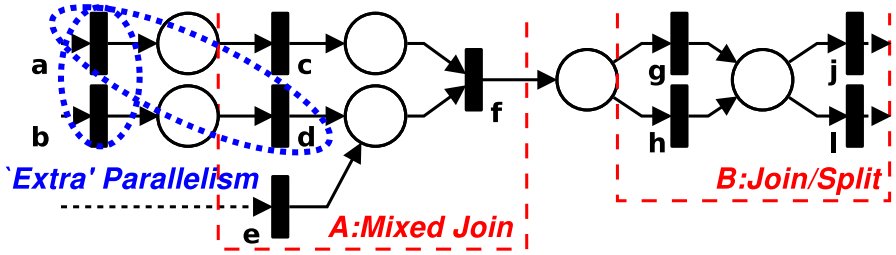


Figure 5.3: Petri Net Fragment showing Complex Splits and Joins (Sub-Structures A and B) and 'Extra' Parallel Activities (Dotted Ellipses).

infeasible, and does not reduce the complexity of the problem.

As discussed above (Theorem 1), we can instead treat sub-structures as built from independent Alpha relations rather than from individual sub-strings. Three further areas then need to be considered in analysing full models.

5.3.1 Compound Splits/Joins

A single Workflow net sub-structure as mined by Alpha may combine both join and split (Figure 5.3 sub-structure B) or combine parallel and exclusive behaviour (sub-structure A). In general, if m paths of which p are XOR (the remainder parallel) join and then split to n paths of which q are XOR, probability of discovery is approximated using the approach of Theorem 1, multiplying probabilities for the relevant $\rightarrow_n$, $\#_n$ and $\|_n$ relations.

5.3.2 Extra Parallelism

Where parallel paths contain more than one activity, such as a, b, c, d in Figure 5.3, for each pair of activities a, b not part of the split or join, either $a \ \|_n \ b$ or $a \#_n b$ must be discovered, to prevent extra dependencies from being inferred. Let $a \nleftrightarrow_n b$ denote $(a \ \|_n \ b) \vee (a \#_n b)$. These are independent (Figure 5.1), so $P_{\alpha,n}(a \nleftrightarrow_n b) = P_{\alpha,n}(a \ \|_n \ b) + P_{\alpha,n}(a \#_n b)$.

5.3.3 Combining Probabilities for Sub-Structures

Let $P_{\alpha,n}(S)$ be the probability of discovering sub-structure S from an event log of n traces. Intuitively, if a split has been mined correctly, then mining the corresponding join is 'almost certain', as each path between the split and join should be in the log. So sub-structures in the model can be considered as dependent on 'previous' sub-structures. For example,

$$P_{\alpha,n}(\mathcal{M}) = P_{\alpha,n}(A) \times P_{\alpha,n}(B|A) \times P_{\alpha,n}(C|B) \times P_{\alpha,n}(D|C) \times P_{\alpha,n}(E|D) \times P_{\alpha,n}(F|B,E),$$

where $P_{\alpha,n}(F|B,E)$ indicates the probability of discovering F conditional that B and E have been mined correctly. This affects the formulae from Section 5.2 in two ways. For each event (such as 'see no ab in the log of n traces'),

1. the probabilities of the sub-strings are conditioned by the probabilities of the prefix strings leading up to those sub-strings, i.e. $\pi(ab)$ becomes $\pi(b|{\to}a)$; and
2. we only consider the traces within which those sub-strings are expected to occur.

To illustrate, for Alpha, the formulae for probability of correct mining of the Alpha relations become:

$$\begin{aligned}
P_{\alpha,n}(a >_n b) &= 1 - \Big(1 - \frac{\pi(ab)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)}, \\
P_{\alpha,n}(a \to_n b) &= \Big(1 - \frac{\pi(ba)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)} - \Big(1 - \frac{\pi(ab)}{\pi(\to a)} - \frac{\pi(ba)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)}, \\
P_{\alpha,n}(a \parallel_n b) &= 1 - \Big(1 - \frac{\pi(ab)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)} - \Big(1 - \frac{\pi(ba)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)} + \\
&\qquad \Big(1 - \frac{\pi(ab)}{\pi(\to a)} - \frac{\pi(ba)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)}, \text{ and} \\
P_{\alpha,n}(a \#_n b) &= \Big(1 - \frac{\pi(ab)}{\pi(\to a)} - \frac{\pi(ba)}{\pi(\to a)}\Big)^{n\cdot\pi(\to a)}.
\end{aligned}$$

This has the effect of a modest reduction in the probability $P_{\alpha,n}(S)$ of successfully mining structure S in n traces. The predicted number of traces n can also be obtained by

using the 'local' probabilities within S, and dividing by the probability of traces 'reaching' the structure. We discuss this further in Appendix B.

5.4 Step 3: Analysis of Alpha Algorithm

We next explore some of the behaviour of Alpha shown by the probability formulae. In this section we refer to 'noise' in event logs. Informally, we define 'noise-free' logs as those recorded without error, by a process which is followed without error. For example, when considering the relation $P_{\alpha,n}(a \rightarrow_n b)$, a noise-free log contains no instances of sub-string ba. Conversely we may specify a 'noise' probability, e.g. $\pi(ba) = 0.1$. We propose a formal definition of noise in Chapter 8.

Basic Relations

Figure 5.4 shows $P_{\alpha,n}(a \rightarrow_n b)$ increasing sharply with increasing $\pi(ab)$ (probability of trace including string ab), but reducing sharply with any probability of the 'reverse' string, $\pi(ba)$. The effect is stronger as n increases, since this also increases the chance of at least one ba in the log. Non-zero probability of ba may be due to errors in logging, or indicate that the real relation is parallel but with ab more likely than ba. Figure 5.5 shows the behaviour when the event log is 'noise-free', i.e. $\pi(ba) = 0$. The probability of correct mining then increases rapidly as either $\pi(ab)$ or n increases.

Relation $a \#_n b$ means a and b are unrelated. Figure 5.6 shows that the probability of this being inferred reduces very quickly as n increases, if either $\pi(ab)$ or $\pi(ba)$ is non-zero.

In Figure 5.7, the parallel relation $a \parallel_n b$, for which both ab and ba must be seen, is seen to be most likely when the probability of either order is similar (note that in Figure 5.7, $\pi(ba) = 0.4 - \pi(ab)$). This is important, since when multiple activities are allowed to occur in any order (parallel), in practice certain orderings may be more likely, reducing the probability of discovering the true parallelism, and necessitating more data.

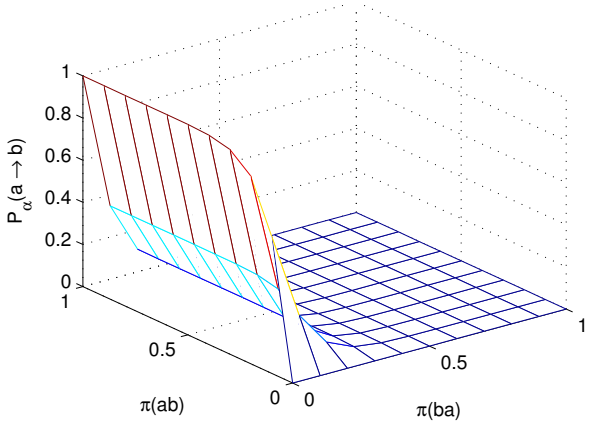


Figure 5.4: Probability of Mining (by the Alpha Algorithm) of $a \rightarrow_n b$ for 10 Traces, varying $\pi(ab)$, $\pi(ba)$.

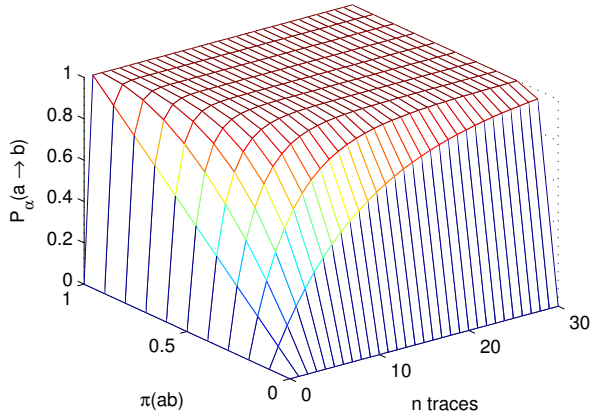


Figure 5.5: Probability of Mining of $a \rightarrow_n b$ for $\pi(ba) = \pi(ab \wedge ba) = 0$, i.e. from Noise-Free Logs. Varying n (Traces) and $\pi(ab)$.

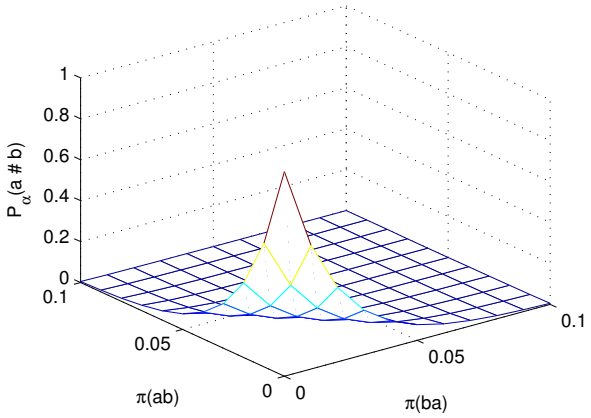


Figure 5.6: Probability of Mining $a \#_n b$, for 50 Traces, varying $\pi(ab)$, $\pi(ba)$.

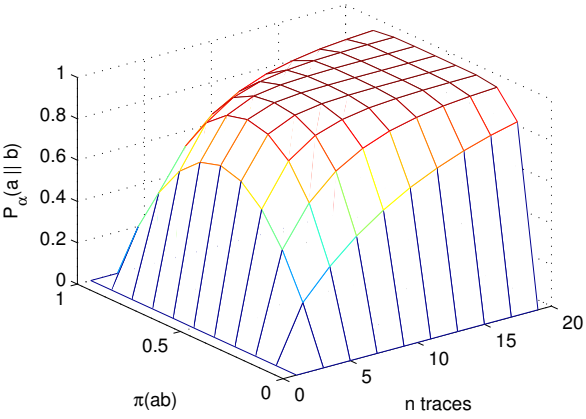


Figure 5.7: Probability of Mining of $a \parallel_n b$ for varying n and $\pi(ab)$, $\pi(ab)+\pi(ba) = 0.4$.

Simple Structures

We next consider the behaviour of Alpha when mining a 3-way XOR split from activity i to activities a, b or c, where $\pi(ia) + \pi(ib) + \pi(ic) = 1$. All possible combinations of these probabilities are indicated by points on the simplex illustrated in Figure 5.8. This simplex is used as the triangular base in the graphs in Figures 5.9–5.14: towards the vertices, one of $\pi(ia), \pi(ib)$ or $\pi(ic)$ approaches one, the other two approach zero; towards the edges, one path probability approaches zero, the sum of the other two approaches one. In the interior of the triangle, all three path probabilities are non-zero but they sum to one.

Figure 5.9 shows the number of traces required for Alpha to achieve 95% probability

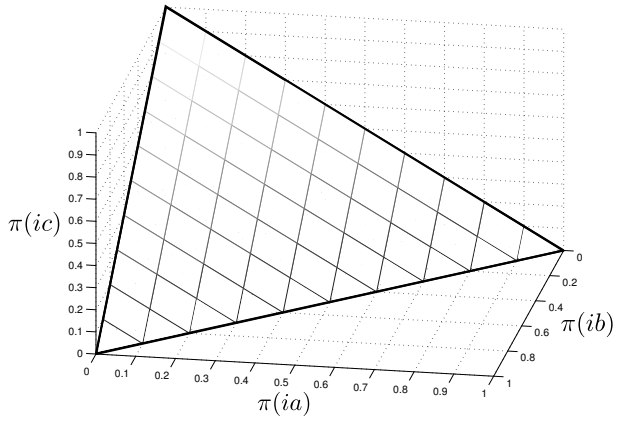


Figure 5.8: Simplex of all Points $\pi_{ia} + \pi_{ib} + \pi_{ic} = 1$, used as the triangular Base of Figures 5.9 – 5.14.

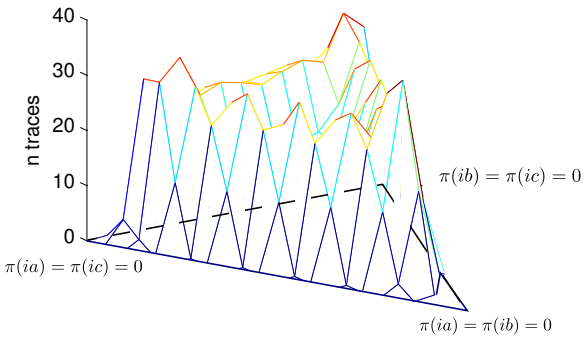


Figure 5.9: Number of Traces for 95% Probability of correct Mining by Alpha of 3-way XOR Split Sub-Structure.

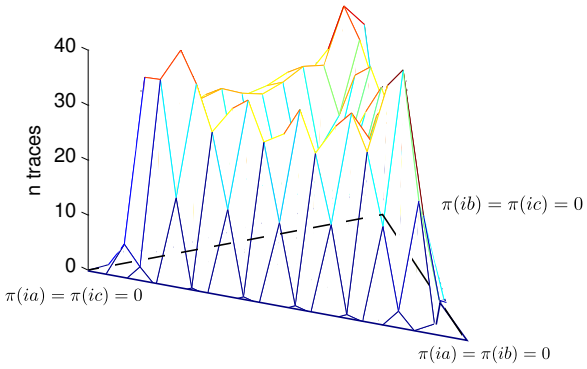


Figure 5.10: Number of Traces for 95% Probability of correct Mining by Alpha of 3-way AND Split Sub-Structure.

of discovery of this XOR split. The greatest number of traces is needed where the probabilities are most imbalanced, i.e. around the edges, with the peaks at each corner showing where only one path has a non-negligible probability.

Figures 5.11–5.14 illustrate the probability of correct mining of the 3-way XOR split as the number of traces vary. The probability is highest when the path probabilities are most evenly split (Figure 5.11), approaching certainty as the number of traces increases (Figure 5.12), except where one or more paths have very low probability. With only a small amount of noise, $\pi(ba) > 0$, with few traces the probability of mining is initially similar (Figure 5.13), but never exceeds 0.6, and rapidly reduces as n increases and the probability increases of seeing at least one ba in the log (Figure 5.14).

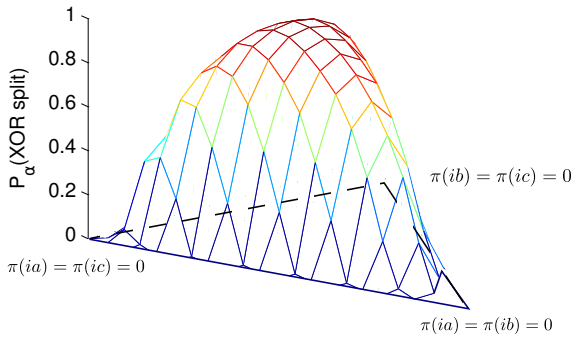


Figure 5.11: Probability of mining of 3-way XOR Split Sub-Structure from 10 Traces (Noise-Free).

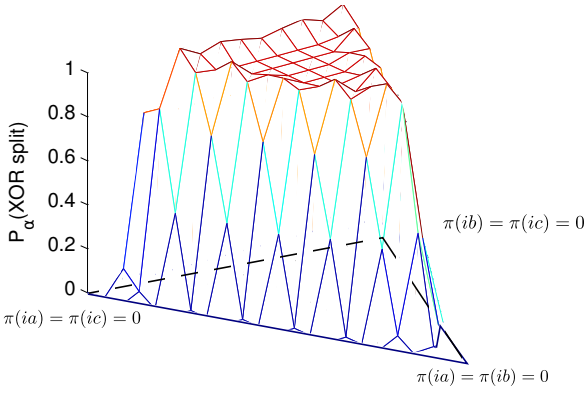


Figure 5.12: Probability of mining of 3-way XOR Split Sub-Structure from 50 Traces (Noise-Free).

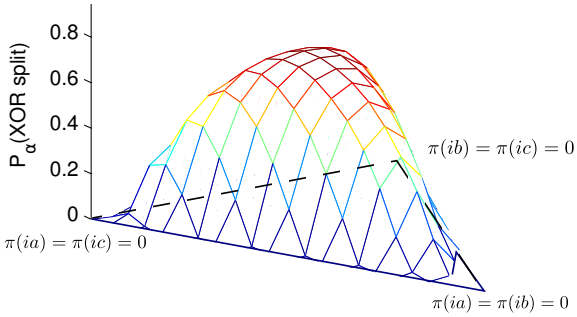


Figure 5.13: Probability of mining of 3-way XOR Split Sub-Structure from 10 Traces With 'Noise' ($\pi(ba) = 0.01$).

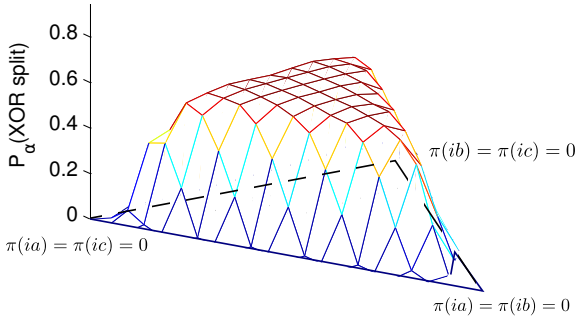


Figure 5.14: Probability of mining of 3-way XOR Split Sub-Structure from 20 Traces With 'Noise' ($\pi(ba) = 0.01$).

Finally, mining of XOR and AND show similar behaviour (Figures 5.9 and 5.10). Around 10% more traces are needed for the AND split than for XOR. In Figure 5.10, points on the triangular base indicate the possible combinations of probabilities of the paths following the parallel split starting with one of the parallel activities. Again, $\pi(ia) + \pi(ib) + \pi(ic) = 1$.

5.4.1 Analysis of Example Process

We used the methods presented to predict the number of traces needed for the probability of successful mining of the running example (Figure 3.1) to exceed various thresholds. Automaton A_0 (Figure 2.5) was specified as the ground truth, encoded using the OpenFst format [9], and randomly walked to produce 30 sets of MXML format [167] event logs of

increasing size from 1 to 45 traces. A 'ground truth' log of 1000 traces was also simulated.

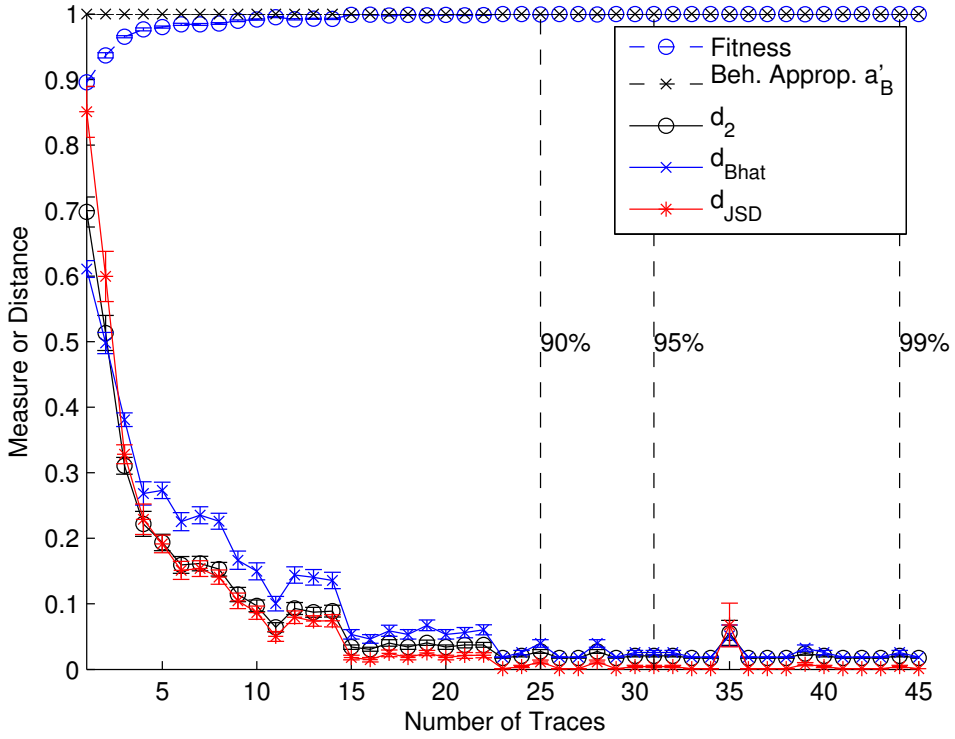
We mined Petri net models from these files using Alpha as implemented in the ProM Framework [170], and calculated the Fitness (f) [124] and Behavioural Appropriateness (a'_B) [12] values using the Conformance Analysis plugin. The Petri nets were converted to PDFA by labelling their reachability graphs with maximum likelihood probability estimates derived from the ground truth log file. The d_2 and Bhattacharyya (d_{Bhat}) distances, and Jensen-Shannon Divergence (d_{JSD}) were calculated between the distributions represented by these PDFA and the ground truth distribution represented by A_0.

The graph in Figure 5.15(a) shows the average *approximate correctness* of the models mined by Alpha from logs of increasing size, as measured by the metrics and distances, plotted against the number of traces in the log. The numbers of traces predicted for $90\%, 95\%$ and 99% confidence in correct mining are indicated by the vertical rules. The graph shows:

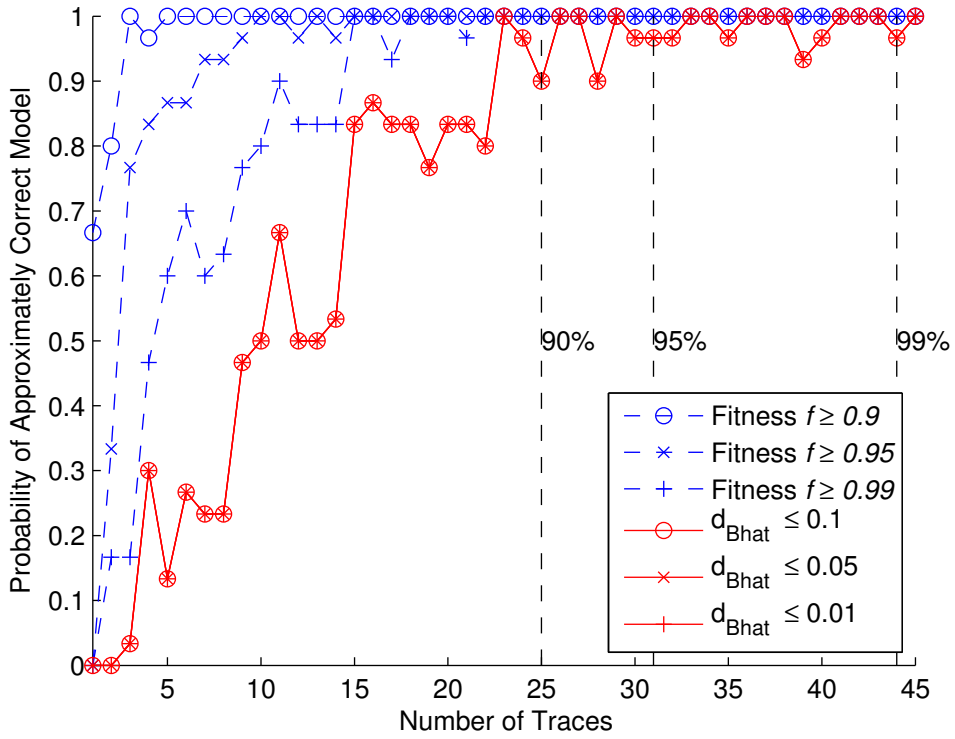
1. Probability distance measures converge in a similar way to f, but the distances from the ground truth are distributed over a clearer scale, from almost 1 for the very unfit models produced by few traces, through to 0, whereas f ranges from approx 0.8 to 1 (see Section 4.3).
2. The distance measures show convergence to approximate correctness at the predicted points.
3. Irregularities may indicate points of interest in the behaviour of the algorithm, worthy of further investigation (see Section 5.4.2).
4. a'_B was 1 for each model, indicating that none of the models allowed behaviour not found in the logs.

Note that close convergence to predictions is possible, because the distribution to be learnt is known in advance, and test data drawn from that distribution. Also, exact formulae rather than bounds are used to predict the numbers of traces.

The graph in Figure 5.15(b) shows the *probability* of mining an approximately correct model, measured by f exceeding $0.9, 0.95$ and 0.99, and d_{Bhat} not exceeding $0.1, 0.05$ and



(a) Average Metrics plotted against Number of Traces



(b) Probability of approximately correct Model.

Figure 5.15: Results showing Convergence of Alpha to the Ground Truth, mining from Logs of increasing size simulated from the Order Process Running Example (PDFA A_0).

Probability of Success	**50%**	**90%**	**95%**	**99%**
Predicted/Actual Traces	11/10	25/23	31/29	44/45
f	0.992	0.998	1.000	1.0
a'_B	1.0	1.0	1.0	1.0
$1 - d_2$	0.935	0.974	0.984	1.0
$1 - d_{Bhat}$	0.866	0.950	0.978	1.0
$1 - d_{JSD}$	0.962	0.991	0.998	1.0

Table 5.1: Metrics and Numbers of Traces at Threshold Points for mining the Order Process Running Example (Petri Net Figure 3.1, PDFA A_0 Figure 2.5).

Probability of Success	**50%**	**90%**	**95%**	**99%**
Predicted/Actual Traces	37/−	63/65	75/75	100/115
f	0.984	0.989	0.998	1.0
a'_B	0.962	0.984	0.998	1.0
$1 - d_2$	0.951	0.983	0.997	1.0
$1 - d_{Bhat}$	0.766	0.881	0.980	1.0
$1 - d_{JSD}$	0.768	0.903	0.983	1.0

Table 5.2: Metrics and Numbers of Traces at Threshold Points for mining Larger Example (Petri Net Figure 5.16, PDFA A_3 Figure 5.17).

0.01. A single data point is calculated for each size log; the percentage of mined models for which f was above, or d_{Bhat} was below the threshold. The probability distance (solid lines) is less sensitive to the threshold used (all three lines are super-imposed), due to operating over a greater range, whereas f (dashed lines) indicates convergence too soon.

Table 5.1 shows the predicted and actual numbers of traces, and corresponding values of the metrics.

5.4.2 Analysis of Large Example Process

The running example is rather simple. To validate the methods and probabilistic analysis of Alpha, we used a larger example (Petri net Figure 5.16, 'ground truth' PDFA Figure 5.17), which permits more detailed analysis and interpretation. This model is a sound WF-net, and so is mineable by Alpha. It shows the handling of a request placed with a technical support call centre. After the call is received (activity i), three streams of

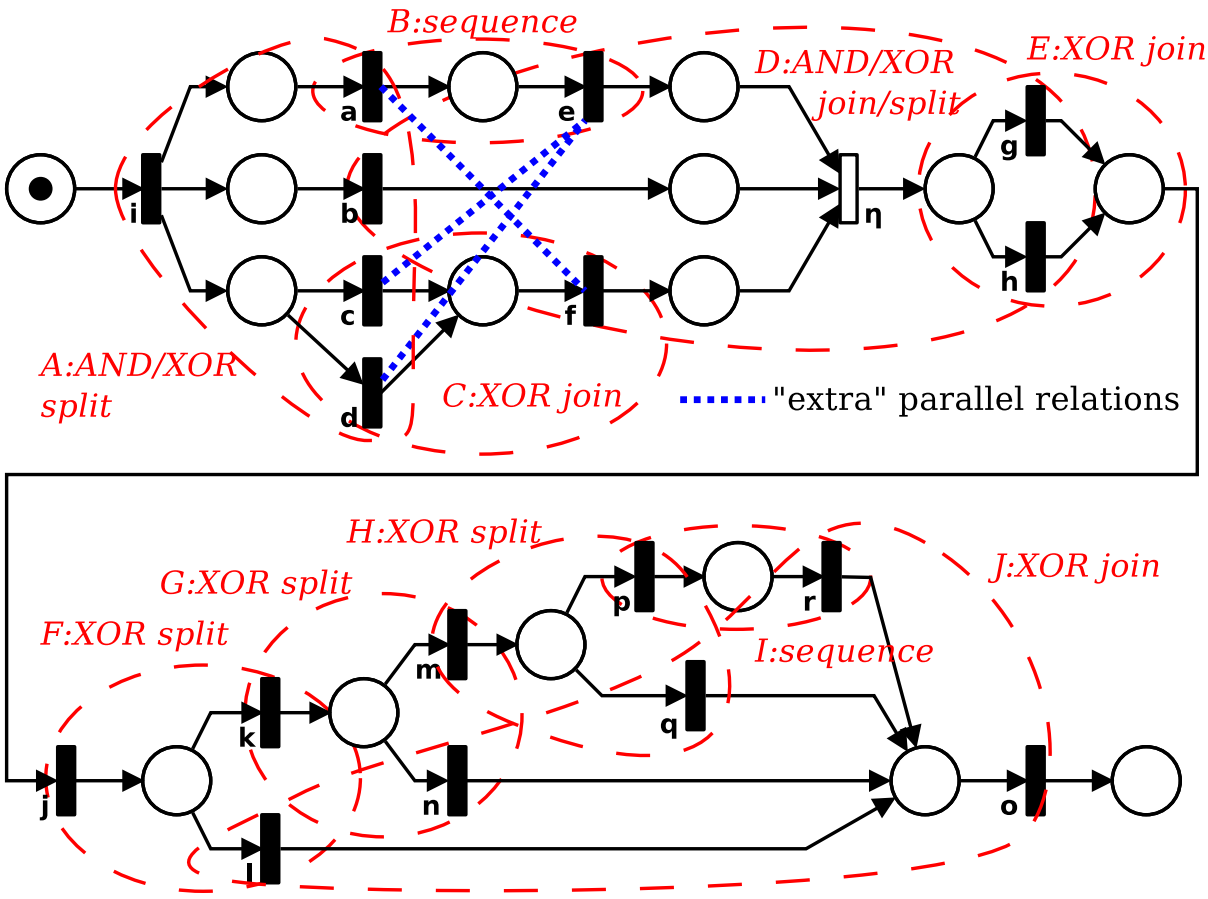


Figure 5.16: Petri Net N_3 representing more complex Example Process with representative Process Sub-Structures and 'Extra' Parallel Relations.

activity run in parallel, synchronised at η[1]. Next either g or h occurs, followed by a series of nested choices (e.g. various actions to resolve the call), before the call is closed (o).

This model was artificially designed as a realistic process with a mix of simple, compound and nested sub-structures, and 'extra' parallelism (parallel activities not part of a split or join sub-structure). Splits in the PDFA were allocated uniform probabilities. The PDFA was simulated to produce event logs from 5 to 150 traces in increments of 5 traces.

Table 5.3 shows for each sub-structure in the model, the number of traces needed for 95% probability of mining the sub-structure correctly. 'Global' indicates the number of traces calculated using the ground truth probabilities for each sub-string, e.g. $\pi(km)$ in the XOR split G. 'Local' gives the number of traces using the local probabilities in

[1]a 'hidden' transition, not recorded in the log, which simplifies the depiction of the net. Alpha produces a behaviourally equivalent net without η.

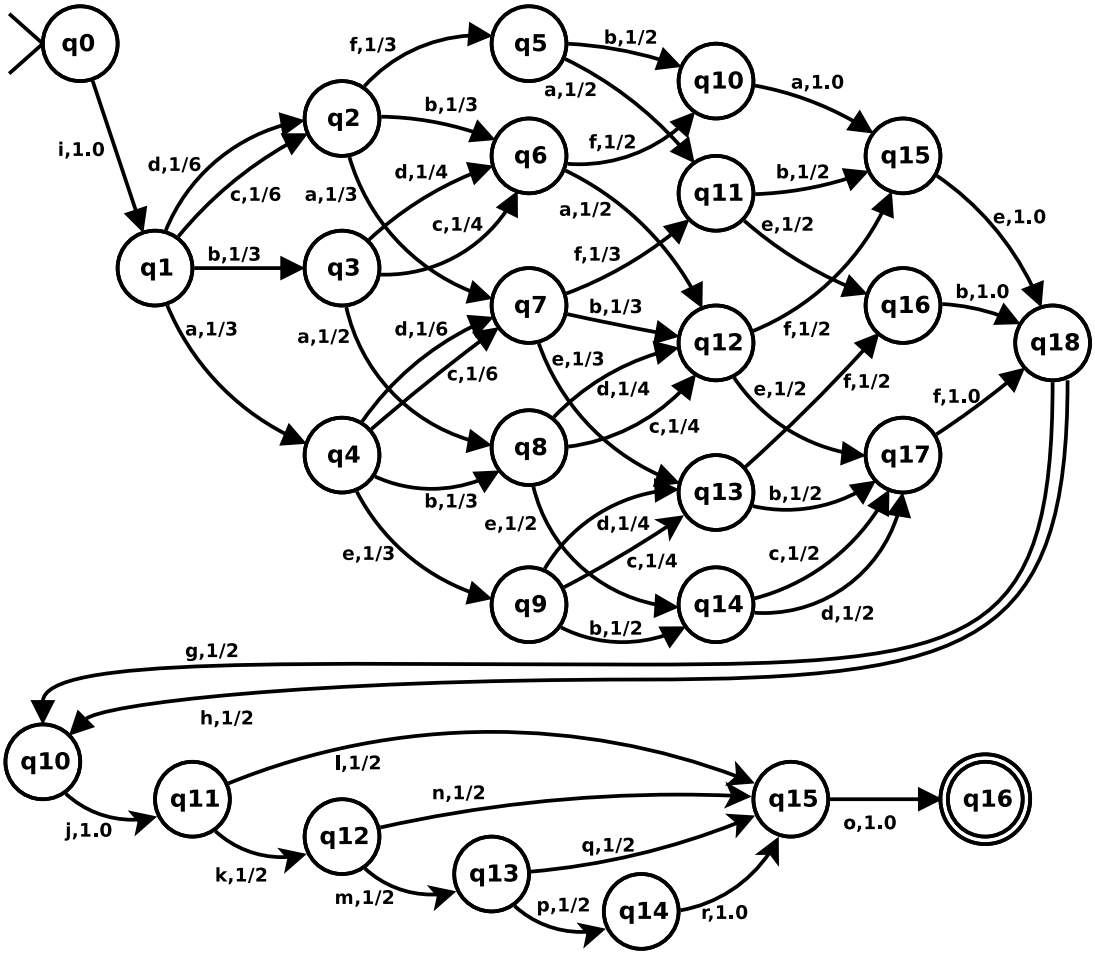


Figure 5.17: PDFA A_3 corresponding to Petri Net N_3, with the addition of Transition Probabilities, used for producing Simulated Event Logs.

each sub-structure, assuming traces that include that part of the model, e.g. $\pi(km|{\rightarrow}k)$. 'Context' shows the number of traces given the sub-structure in its context in the model, i.e. for G, only $\pi({\rightarrow}k)$ of the traces are expected to reach k, so the number of traces estimated in the 'Local' column is divided by $\pi({\rightarrow}k)$.

The graphs (Figure 5.18) again show convergence as predicted (Table 5.2). The shapes of the graphs suggest correspondence with the numbers of traces predicted for discovery of sub-structures (Table 5.3), e.g. the 'plateaus' between $30-35, 40-45$, and $50-55$ traces. By 30 traces $a \nleftrightarrow f$ (and XOR split H) will be mined correctly, with high confidence. By 40 traces all the 'extra parallelism' will be, and by 50 traces A also should be discovered, giving confidence that most of the first (complex) part of the model will be correct. By 60 traces, with 95% confidence all sub-structures will be mined correctly.

Structure	Global	Local	Context
A: AND/XOR split	49	49	49
B: Sequence	5	5	5
C: XOR join	13	**6**	6
$a \nleftrightarrow f, c \nleftrightarrow e, d \nleftrightarrow e$ ‘	$26, 51, 60$	$26, \mathbf{34}, \mathbf{36}$	$26, 34, 36$
D: AND/XOR join/split	56	56	56
E: XOR join	6	**1**	1
F, G, H: XOR splits	$6, 13, 28$	$6, \mathbf{6}, \mathbf{6}$	$6, \mathbf{12}, \mathbf{24}$
I: Sequence	23	**1**	**9**
J: XOR join.	28	**1**	1

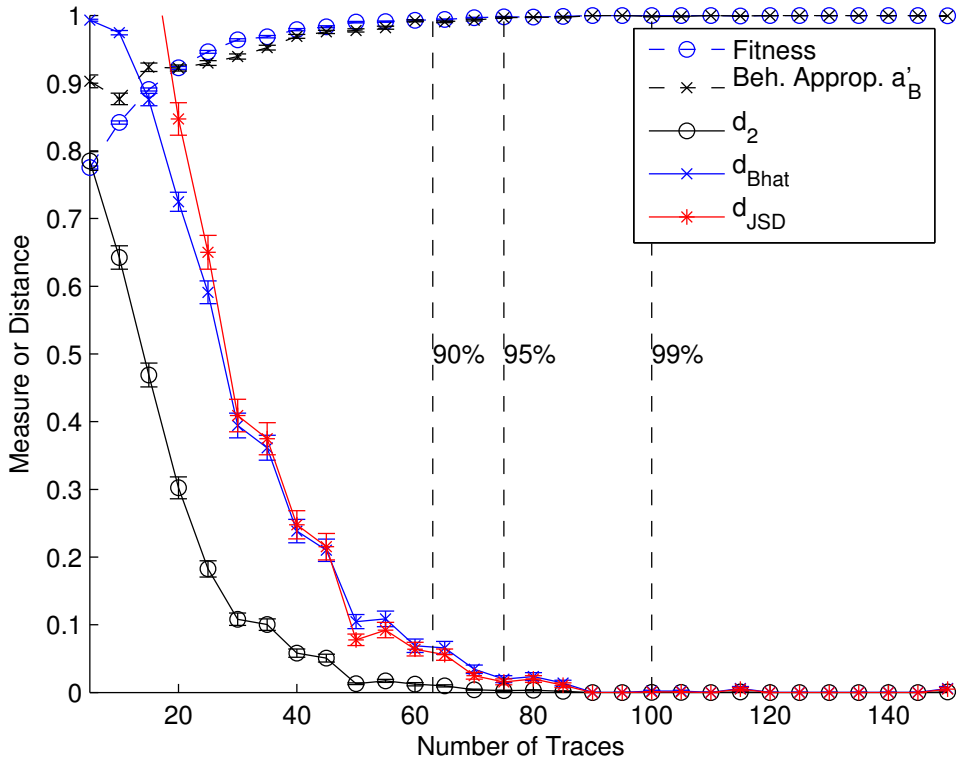
Table 5.3: Predicted Numbers of Traces to mine Sub-Structures in N_3, Bold shows the Minimum Predictions for each Structure.

Between these points ($35, 45, 55$ traces) no additional structures are expected to be mined correctly. Fitness and Behavioural Appropriateness are both below 1 at low numbers of traces, indicating that the mined models do not fit all the traces in the log, and are also too general, allowing behaviour not seen in the log. This is captured in the shape of the distance graphs at low numbers of traces; showing that convergence is initially slow.

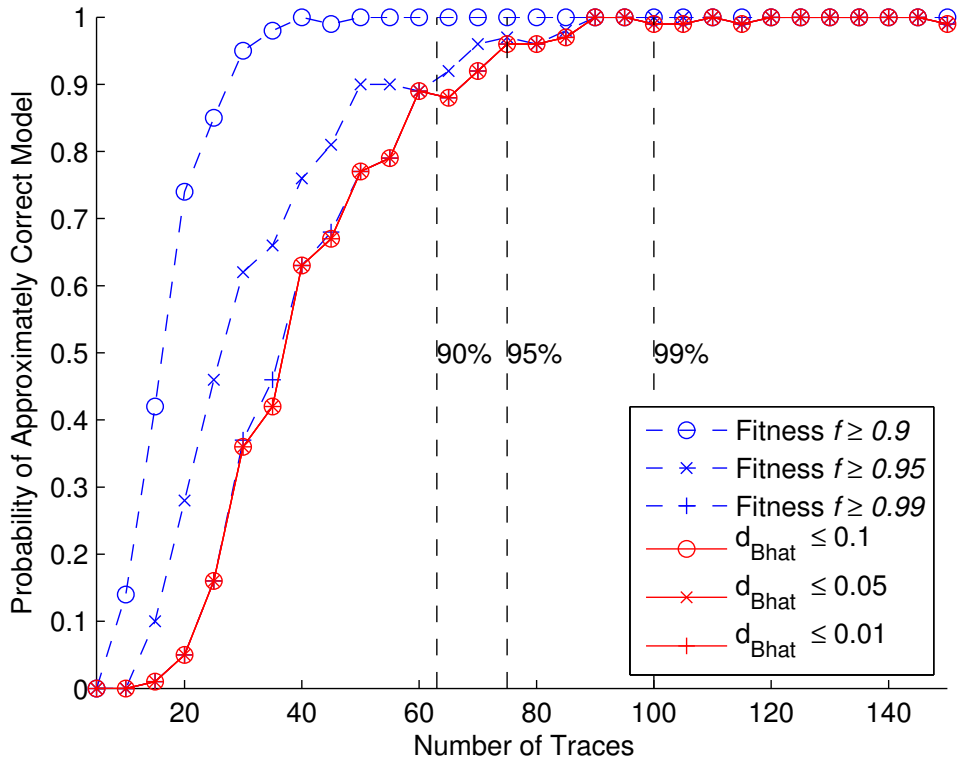
5.5 Chapter Summary

We applied the framework for analysing process mining algorithms (Chapter 4), to the Alpha algorithm [156]. We developed formulae for the probability of correct mining by Alpha of basic process structures, and extended these to a method for predicting the number of process traces needed for confidence in correctly mining arbitrary process models. Experimentation on representative process models showed the accuracy of these predictions, and that the method gives insight into the learning behaviour of the algorithm.

Alpha is relatively simple and makes many assumptions such as correct recording of the traces (no ‘noise’), and that the underlying process can be modelled by a Structured Workflow Net, but the same method can in principle be applied to any process mining algorithm. We next investigate in the same manner a more practically useful algorithm.



(a) Average Metrics plotted against Number of Traces



(b) Probability of approximately correct Model.

Figure 5.18: Results showing Convergence of Alpha to the Ground Truth, mining from Logs of increasing size simulated from Larger Process Model (PDFA A_3).

CHAPTER 6

CASE STUDY: ANALYSIS OF THE HEURISTICS MINER ALGORITHM

In the previous chapter we analysed a simple process mining algorithm, the Alpha Algorithm [156], under our framework for analysis of process mining algorithms introduced in Chapter 4. We showed that by breaking a process model into basic structures, we can use the probabilities in these structures to predict the number of process traces needed for Alpha to correctly mine them, and thus the full model. We showed that using this probabilistic view of process mining gives useful insights into the learning process.

Alpha is an early, simple algorithm, initially created to prove that a certain class of processes could be mined, and to explicitly deal with concurrent activities. In this chapter we apply the framework to the analysis of a more practically useful mining algorithm, the Heuristics Miner [194]. This algorithm is so called because it uses a number of parameters and thresholds to control the complexity of the mined model. The algorithm is quite simple, but as we will see, this masks complex probabilistic behaviour.

An abridged version of the material in this chapter was first presented in [189].

6.1 A Probabilistic Analysis of the Heuristics Miner

The Heuristics Miner algorithm was described in Chapter 3. The key to the behaviour of the algorithm is the Dependency Measure (Equation 3.1, Chapter 3) (DM), used to

indicate the strength of causal relationship between pairs of activities, and thus which arcs to include in the model. We first investigate the behaviour of the DM, then how it is used to construct basic process structures.

6.1.1 The Dependency Measure

Since we assume no cycles, a sub-string $w \in \Sigma^+$ occurs zero or one times in any trace in $\mathcal{W}$, with probability $\pi(w)$. $N(w)$, the number of times w occurs in event log $\mathcal{W}$ of n traces, is Binomially distributed (Chapter 4),

$$Q_n\big(N(w)\big) = Bin\big(\pi(w), n\big).$$

We also write $Q_n\big(N(w), N(v)\big)$, where $v \in \Sigma^+$, for the joint probability of $N(w)$ and $N(v)$ in the event log, and $Q_n\big(N(w)|N(v)\big)$ for the conditional probability of $N(w)$ occurrences of sub-string w, given $N(v)$ occurrences of sub-string v in $\mathcal{W}$.

The DM (3.1) between two activities a and b is a random variable which we can write as a function τ of two such counts obtained from $\mathcal{W}$,

$$\mathrm{DM}_{ab} = \tau\big(N(ab), N(ba)\big) = \frac{N(ab) - N(ba)}{N(ab) + N(ba) + 1}. \tag{6.1}$$

The expected value of DM_{ab} obtained from a log is with respect to $Q_n\big(N(ab), N(ba)\big)$,

$$\mathbb{E}_{Q_n(N(ab),N(ba))}[\mathrm{DM}_{ab}] = \sum_{N(ab)=0}^{n} \sum_{N(ba)=0}^{n} Q_n\big(N(ab), N(ba)\big)\tau\big(N(ab), N(ba)\big).$$

By simulation and numerical integration it can be demonstrated that

$$\begin{aligned}\mathbb{E}_{Q_n(N(ab),N(ba))}[\mathrm{DM}_{ab}] &= \mathbb{E}_{Q_n(N(ab),N(ba))}\Big[\frac{N(ab) - N(ba)}{N(ab) + N(ba) + 1}\Big] \\ &= \frac{\mathbb{E}_{Q_n(N(ab))}[N(ab)] - \mathbb{E}_{Q_n(N(ba))}[N(ba)]}{\mathbb{E}_{Q_n(N(ab))}[N(ab)] + \mathbb{E}_{Q_n(N(ba))}[N(ba)] + 1} = \frac{n\pi(ab) - n\pi(ba)}{n\pi(ab) + n\pi(ba) + 1},\end{aligned}$$

i.e. expected value of DM_{ab} can be obtained from expected values of the sub-string counts.

Activities Which Occur in One Order Only

If $\pi(ai) = 0$ and $\pi(ia) \in]0, 1[$, then activities i and a can only occur in one order, i.e. only ia may be seen in event log $\mathcal{W}$. $Q_n\big(N(ia), N(ai)\big) = 0$ for $N(ai) \neq 0$ and

$$\mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}] = \sum_{N(ia)=0}^{n} Q_n\big(N(ia)\big)\tau\big(N(ia), 0\big) = \frac{n\pi(ia)}{n\pi(ia) + 1}.$$

$\mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}]$ increases monotonically with growing n, from 0 to 1, e.g.

$$\mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}] = 0, \text{ when } n = 0,$$
$$\lim_{n\to\infty} \mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}] = \frac{\pi(ia)}{\pi(ia)} = 1.$$

Activities Which Occur in Either Order

Next consider two activities a and b which may occur in either order (which Heuristics Miner interprets as occurring in parallel). Without loss of generality let $0 < \pi(ab) \leq \pi(ba) < 1$. Since we do not consider cycles, ab and ba cannot occur together in a trace, and $\pi(ab) + \pi(ba) \leq 1$. Then $\mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}]$ converges to some $d \in [0, 1]$ as n increases:

$$\mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}] = 0, \text{ when } n = 0, \text{ or } n > 0 \text{ and } \pi(ab) = \pi(ba),$$
$$\lim_{n\to\infty} \mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}] = \frac{\pi(ba) - \pi(ab)}{\pi(ba) + \pi(ab)} = d \in [0, 1].$$

For such DM_{ia} and DM_{ba} there will always exist some number of traces $n' \geq 0$ above which $\mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}] > \mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}]$ such that for a log of n traces,

$$\text{if } n \leq n' \text{ then } \mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}] \leq \mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}],$$
$$\text{if } n > n' \text{ then } \mathbb{E}_{Q_n(N(ia),0)}[\mathrm{DM}_{ia}] > \mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}].$$

Since Dependency Measures are random variables, we are not interested in n' but

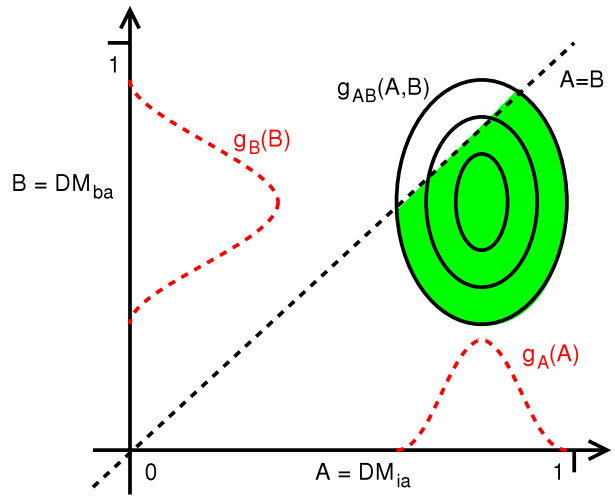


Figure 6.1: Illustration of Dependency Measures $A = \mathrm{DM}_{ia}$, $B = \mathrm{DM}_{ba}$ and their Marginal Distributions g_A, g_B and Joint Density g_{AB}. Shaded Area illustrates $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ab})$.

probabilities such as $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$, that DM_{ia} obtained from a given event log $\mathcal{W}$ exceeds DM_{ba} from the same $\mathcal{W}$. This is represented by the shaded area in Figure 6.1. As a shorthand let $A = \mathrm{DM}_{ia}$, $B = \mathrm{DM}_{ba}$, and let g_A be the Probability Density Function (PDF) of A, g_B the PDF of B, and g_{AB} the joint density. Likewise G_A (etc.) for the corresponding Cumulative Distribution Function (CDF),

$$G_A(A) = \int_{t=-\infty}^{t=A} g_A(t)\, dt.$$

Then

$$\gamma_n(A > B) = \int_{A=-\infty}^{A=+\infty} \int_{B=-\infty}^{B=A} g_{AB}(A, B)\, dA\, dB.$$

If DM_{ia} and DM_{ba} are independent,

$$\gamma_n(A > B) = \int_{A=-\infty}^{A=+\infty} \int_{B=-\infty}^{B=A} g_A(A) g_B(B)\, dA\, dB = \int_{A=-\infty}^{A=+\infty} g_A(A) G_B(A)\, dA. \qquad (6.2)$$

Probability Distributions for Dependency Measures

We next investigate the form of the probability distributions followed by Dependency Measures. This will be used in subsequent sections in considering the requirements for correctly mining acyclic process structures (such as sequences of activities, XOR or AND splits and joins), including correct ordering of DMs between activities within structures.

To simplify notation, let $X = N(ba)$, $Y = N(ab)$, $Z = X - Y$, $W = X + Y + 1$, then the Dependency Measure (DM),

$$\mathrm{DM}_{ba} = \frac{N(ba) - N(ab)}{N(ba) + N(ab) + 1} = \frac{X - Y}{X + Y + 1} = \frac{Z}{W}, \tag{6.3}$$

follows a discrete distribution which is the ratio of two random variables Z and W. Z is the difference between two Binomial random variables X and Y, W the sum of X and Y. The distribution of DM_{ba} is complex and difficult to model analytically. However, if we assume the conditions are satisfied to approximate X and Y by Gaussians, then Z and W can also be approximated by Gaussians and we can model the distribution of the Dependency Measure as the ratio of two Gaussians, following the framework of Marsaglia [100, 101] and Cedilnik *et al.* [34].

Marsaglia shows that the ratio of arbitrary Gaussian random variables follows a distribution which is the product of a standard centred Cauchy distribution and a bimodal distribution, and is itself difficult to handle analytically [100]. Marsaglia presents empirical analysis of the types of distributions which result from ratio variables with various different means and variances. It turns out that for the variables which we encounter in the Dependency Measures (means in the range $[-1, 1]$), the distribution will have only one significant mode. The distributions of the Dependency Measures vary from approximately Gaussian in shape, to significantly skewed. Figure 6.2 shows two examples.

Let DM_{ba} be a Dependency Measure with CDF $G_{ba}(t) = \gamma_n(\mathrm{DM}_{ba} \leq t)$, PDF $g_{ba}(t) = G'_{ba}(t)$. X and Y are outcomes of a Multinomial distribution, which we marginalise and approximate with Gaussians with means $\mu_X = n\pi(ba), \mu_Y = n\pi(ab)$ and variances

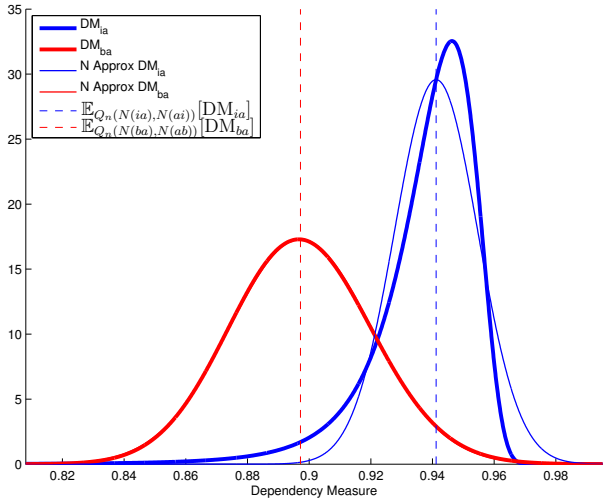


Figure 6.2: Example DM Distributions DM_{ia} ($\pi(ia) = 0.05$, $\pi(ai) = 0$), DM_{ba} ($\pi(ba) = 0.95$, $\pi(ab) = 0.05$), $n = 320$, also showing Gaussian Approximation (Section 6.2.5).

$v_X = n\pi(ba)\big(1 - \pi(ba)\big), v_Y = n\pi(ab)\big(1 - \pi(ab)\big)$ respectively. Let μ_z, μ_w and v_z, v_w be the means and variances of the numerator (Z) and denominator (W) of the DM:

$$\mu_z = \mu_X - \mu_Y, \qquad \mu_w = \mu_X + \mu_Y + 1,$$
$$v_z = v_X + v_Y - 2cov(X,Y), \qquad v_w = v_X + v_Y + 2cov(X,Y),$$

where $cov(X,Y) = n\pi(ab)\pi(ba)$ is the covariance between X and Y given by the Multinomial distribution. We also require the covariance between Z and W:

Lemma 1. *The covariance c between Z and W is $c = v_X - v_Y$.*

Proof. Taking the expectations with respect to distribution $Q_n\big(N(ba), N(ab)\big)$,

$$\begin{aligned} c &= \mathbb{E}[\big(Z - \mathbb{E}[Z]\big)\big(W - \mathbb{E}[W]\big)] \\ &= \mathbb{E}[\big(X - Y - \mathbb{E}[X - Y]\big)\big(X + Y + 1 - \mathbb{E}[X + Y + 1]\big)]. \end{aligned}$$

Multiplying out and simplifying leads to

$$\begin{aligned} c &= \mathbb{E}[X^2 - Y^2 - 2X\mathbb{E}[X] + 2Y\mathbb{E}[Y] + \big(\mathbb{E}[X]\big)^2 - \big(\mathbb{E}[Y]\big)^2] \\ &= \mathbb{E}[\big(X - \mathbb{E}[X]\big)^2 - \big(Y - \mathbb{E}[Y]\big)^2] = v_X - v_Y. \end{aligned}$$

□

Then following [100], the CDF and PDF of the DM are

$$G_{ba}(t) = \Phi\Big(\frac{t\mu_w - \mu_z}{\sqrt{v_z - 2tc + t^2 v_w}}\Big), \text{ and}$$
$$g_{ba}(t) = G'_{ba}(t) = \phi\Big(\frac{t\mu_w - \mu_z}{\sqrt{v_z - 2tc + t^2 v_w}}\Big)\frac{\mu_w v_z - c\mu_z + (\mu_z v_w - c\mu_w)t}{(v_z - 2tc + t^2 v_w)^{\frac{3}{2}}}, \tag{6.4}$$

where

$$\phi(t) = \frac{e^{-\frac{1}{2}t^2}}{\sqrt{2\pi}}, \quad \Phi(x) = \int_{-\infty}^{x} \phi(t)dt = \frac{1}{2}\Big[1 + \text{erf}\Big(\frac{t}{\sqrt{2}}\Big)\Big]. \tag{6.5}$$

Figure 6.2 illustrates the distributions followed by two DMs, DM_{ia} and DM_{ba}. The taller peak is DM_{ia} (mean tends to 1 with increasing n), the wider peak is DM_{ba} (mean tends to 0.9). Equations (6.4, 6.5) have some similarity with the formula for a Gaussian centred at the expected value of the DM, but with variance dependent on t, which explains the skew of the distribution. The variance also depends on n via the variances of the Binomial random variables X and Y. As n increases, not only do the DMs tend to their limiting values (from an infinitely large event log), but their variances reduce. This has the effect of 'separating' the two distributions, concentrating the joint distribution to one side of the $\text{DM}_{ia} = \text{DM}_{ba}$ line (see Figures 6.4, 6.6 later). The lighter curves are Gaussian approximations to the distributions (Section 6.2.5), illustrating varying skew of distributions of different DMs.

The formulae for the distributions (6.4) are difficult to work with. The joint distribution even for two Dependency Measure variables, even assuming independence (6.2), cannot be integrated analytically.

Often, independence between the Dependency Measures cannot be assumed, and further approximations will be needed. In the next subsections we introduce methods and approximations for obtaining the probability of correct mining by Heuristics Miner of basic process structures, from noise-free logs. We consider only acyclic structures: sequences, exclusive-OR (XOR) and parallel (AND) splits.

6.2 Basic Process Structures

Business processes tend to be well structured, see e.g. [133,154]. In Chapter 4 we described representations of acyclic process structures in our probabilistic framework. In this section we discuss the requirements for Heuristics Miner to correctly mine these basic acyclic process structures, assuming noise-free logs.

6.2.1 Sequences

As with the Alpha algorithm (Chapter 5), if activities a and b form a sequence in the model then if a occurs, it is immediately followed by activity b, and no other. In the simplest case that no other activity c can occur in parallel with a or b, neither ac nor cb is possible in $\mathcal{W}$. So if at least one trace contains ab, DM_{ab} will be the only positive DM in the row for a and the column for b in the Dependency Matrix. No traces will include ba. since we assume no noise. The UH parameter ensures arc $a \rightarrow b$ is created, regardless of the other parameters. The probability of discovery of the sequence (over n traces) is therefore the complement of the probability that every trace in $\mathcal{W}$ will not contain ab,

$$P_{HM,n}(a \rightarrow_n b) = 1 - \big(1 - \pi(ab)\big)^n. \tag{6.6}$$

If an earlier XOR split means the sequence is not always executed, then $\pi(ab) < 1$. If other parts of the model may execute in parallel, then other activities may 'interfere' in recording ab in $\mathcal{W}$, e.g. acb might be recorded, reducing $\pi(ab)$. This suggests that other structures may be mined instead of sequence ab, such as a split from $a \rightarrow (x \,\#\, b)$, and that (6.6) is an oversimplification. We return to this in Chapter 8, Section 8.2.

6.2.2 Splits and Joins

In the next sub-sections we consider mining exclusive and parallel splits and joins. Recall from Section 3.2.2 that Heuristics Miner uses an 'AND measure' and corresponding

threshold to differentiate exclusive (XOR) and parallel (AND) splits and joins. In this chapter we do not consider these as they are only involved when event logs are highly 'noisy' (i.e. a large proportion of the traces in the log do not truly represent the underlying process). When we consider noise-free logs (this chapter) or limited amounts of noise (Chapter 8), the main requirement for determining the types of splits and joins is rather that the Dependency Measures are ordered correctly, ensuring that the correct causal links (those supported by the underlying model) are re-created in the Dependency Graph.

6.2.3 Exclusive (XOR) Splits and Joins

An m-way XOR split occurs where there is a choice between m mutually exclusive paths through the model after activity a, each path starting with an activity $b' \in \{b_i | 1 \leq i \leq m\}$. It is specified in a similar way as for Alpha. Similarly to discovery of a sequence, at least one trace in $\mathcal{W}$ must contain ab_1, ab_2. and so on. Since there is no noise, we may assume $\pi(b_1 a) = \pi(b_2 a) \ldots \pi(b_m a) = 0$, and likewise none of the activities b_i will occur together in a trace, i.e. $\pi(b_i b_j) = 0, \forall\, 1 \leq i, j \leq m$. Each b_i can only have a as a predecessor, so the PO, RTB and DT parameters are again not involved. Therefore in a similar way as for Alpha (Section 5.2.4), the probability of discovery of the split from a log of n traces is

$$P_{HM,n}\big(a \rightarrow_n (b_1 \,\#\, \ldots \,\#\, b_m)\big) = 1 - \sum_{1 \leq i \leq m} \big(1 - \pi(ab_i)\big)^n + \sum_{1 \leq i < j \leq m} \big(1 - \pi(ab_i) - \pi(ab_j)\big)^n - \ldots + (-1)^m \big(1 - \sum_{1 \leq i \leq m} \pi(ab_i)\big)^n. \tag{6.7}$$

Joins are treated in the same way.

6.2.4 2-Way Parallel Splits (AND2)

A two way parallel split ('AND2', Figure 6.3a) occurs where two paths through the model proceed in parallel, following activity i. Let one path start with activity a, the other with

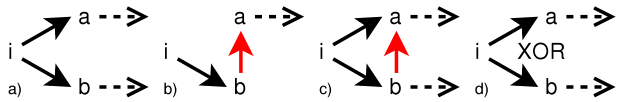


Figure 6.3: a) AND2 True Structure, and Failures Due to b) Missing Path, c) Extra Arc, or d) Interpreting as XOR Structure.

b. We consider first the simplest case, where no other parts of the model occur in parallel, i.e. $\pi(ia) + \pi(ib) = \pi(\rightarrow i) \in [0, 1]$, ignoring any activities that follow a, b in the parallel paths. Only two part traces, iab, iba, are possible and $N(ab) = N(ia)$, $N(ba) = N(ib) = m - N(ia)$, where $m = N(ia) + N(ib)$. With no noise, $N(ai) = N(bi) = 0$. So

$$\mathrm{DM}_{ia} = \tau\big(N(ia), 0\big) = \frac{N(ia)}{N(ia)+1}, \tag{6.8}$$

$$\begin{aligned}\mathrm{DM}_{ba} &= \tau\big(N(ba), N(ab)\big) = \tau\big(m - N(ia), N(ia)\big) \\ &= \frac{\big(m - N(ia)\big) - N(ia)}{(m - N(ia)) + N(ia) + 1} = \frac{m - 2N(ia)}{m+1}.\end{aligned} \tag{6.9}$$

Without loss of generality we assume $\pi(ib) > \pi(ia)$, so $\mathbb{E}_{Q_n(ab,ba)}[\mathrm{DM}_{ab}] < 0$ and we can assume $\mathrm{DM}_{ib} > \mathrm{DM}_{ab}$.

Consider sub-matrix M_S of Dependency Matrix M relevant to activities i, a, b, and how its elements must relate to each other to correctly discover the arcs for the AND2 split. As the number n of traces in $\mathcal{W}$, increases,

$$\lim_{n\to\infty} M_S = \begin{array}{c} \text{activity} \\ i \\ a \\ b \end{array} \begin{array}{c} \begin{array}{ccc} i & a & b \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ -1 & 0 & -P \\ -1 & P & 0 \end{pmatrix} \end{array},$$

in which $P = \lim_{n\to\infty} \mathbb{E}_{Q_n(N(ba),N(ia))}[\mathrm{DM}_{ba}] \in [0, 1]$.

To correctly mine (Figure 6.3(a)) the split from the log we require

1. $\mathrm{DM}_{ia} > \mathrm{DM}_{ba}$: otherwise DM_{ba} would be the largest in the a column of the depen-

dency matrix, and b would be chosen instead of i as the predecessor of a[1]:

$$\text{if } M_S = \begin{array}{c} \text{activity} \\ i \\ a \\ b \end{array} \begin{array}{c} \begin{array}{ccc} i & a & b \end{array} \\ \begin{pmatrix} 0 & \mathrm{DM}_{ia} < \mathrm{DM}_{ba} & \to 1 \\ \text{n/a} & 0 & -\mathrm{DM}_{ba} \\ \text{n/a} & \mathrm{DM}_{ba} > \mathrm{DM}_{ia} & 0 \end{pmatrix} \end{array}$$

then the mined model will contain iba in sequence (Figure 6.3(b)), and not support the alternate order iab.

2. Either $\mathrm{DM}_{ia} > \mathrm{DM}_{ba} + \mathrm{RTB}$, $N(ba) < \mathrm{PO}$ or $\mathrm{DM}_{ba} < DT$ (see Equations (3.2) – (3.6)): otherwise extra arc $b \to a$ will be retained (Figure 6.3(c)).
3. Both $N(ia) > \mathrm{PO}$ and $N(ib) > \mathrm{PO}$: otherwise the split will be XOR rather than parallel (Figure 6.3(d)).

We now consider estimating the probability of these requirements being met, given event log $\mathcal{W}$ of n traces. From (6.8)–(6.9), DM_{ia} and DM_{ba} both depend on $N(ia)$ only, so DM_{ba} is functionally dependent on DM_{ia} and the joint distribution of DM_{ia} and DM_{ba} lies on a line, illustrated in Figure 6.4. We show in proposition 7 that the dependency is negative, i.e. DM_{ba} decreases with increasing DM_{ia}, so the slope of the curve is negative.

Proposition 7. DM_{ba} *is monotonically decreasing with increasing* DM_{ia}.

Proof. Differentiating DM_{ia} with respect to $N(ia)$ shows that DM_{ia} increases with $N(ia)$.

$$\frac{\partial\, \mathrm{DM}_{ia}}{\partial N(ia)} = \frac{1}{\big(N(ia)+1\big)^2} > 0 \text{ for } N(ia) > 0.$$

Similarly, DM_{ba} decreases with increasing n, independent of $N(ia)$:

$$\frac{\partial\, \mathrm{DM}_{ba}}{\partial N(ia)} = \frac{-2}{n+1} < 0.$$

[1]Recall that using 'Use All-Activities-Connected' Heuristic (UH), a predecessor and successor will be chosen for each activity, except for the start and end activities.

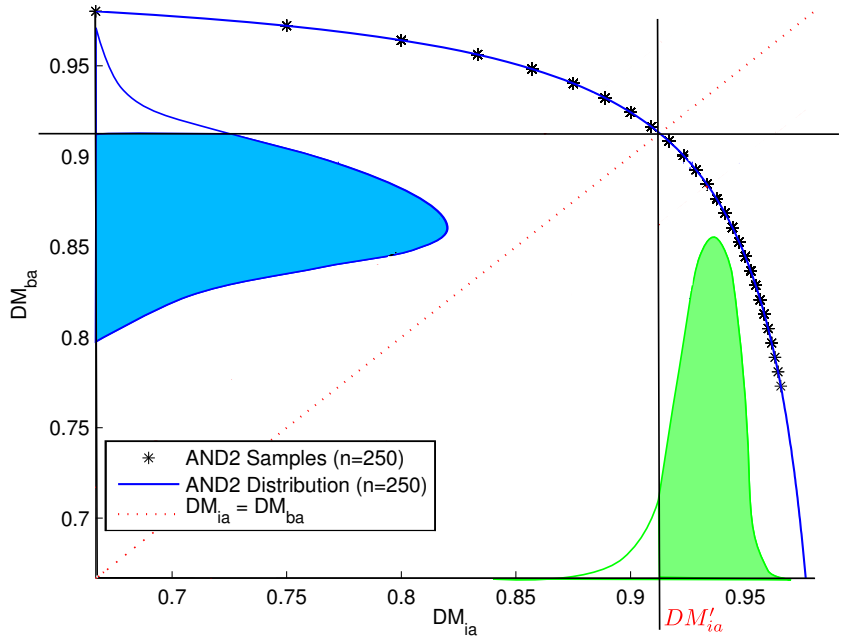


Figure 6.4: Illustration of DM_{ba} plotted against DM_{ia}, for Samples from an Example 'AND2' Distribution ($\pi(ia) = 0.05$, $\pi(ib) = 0.95$, $n = 250$ Traces). The Overlay illustrates the Curve on which the Distribution lies, and the Areas under the Marginal DM_{ia} and DM_{ba} Distributions for which $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$ (Equation 6.12).

Therefore as $N(ia)$ increases, DM_{ia} increases and DM_{ba} decreases, so DM_{ba} decreases with increasing DM_{ia}, i.e. the slope of the curve is always negative. □

Thus we can calculate $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$ by equating the right hand sides of Equations (6.8) and (6.9) to obtain the value $N(ia)'$ of $N(ia)$ for which $\mathrm{DM}_{ia} = \mathrm{DM}_{ba}$,

$$\begin{aligned} \frac{N(ia)'}{N(ia)'+1} &= \frac{m-2N(ia)'}{m+1} \\ \Rightarrow N(ia)'(m+1) &= \big(N(ia)'+1\big)\big(m-2N(ia)'\big) \\ \Rightarrow 2N(ia)'^2 + 3N(ia)' - m &= 0, \end{aligned} \tag{6.10}$$

which has only one valid solution for $N(ia)' \in [0, m]$, for which the DMs are equal,

$$N(ia)' = \tfrac{1}{4}\Big(-3 + \sqrt{8m+9}\Big).$$

For every realisation of $N(ia)$ which is greater than $N(ia)'$, $\mathrm{DM}_{ia} > \mathrm{DM}_{ba}$, i.e.

$$\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) = \sum_{N(ia)>N(ia)'}^{n} Q_n\big(N(ia)\big). \tag{6.11}$$

Since DM_{ia} increases with $N(ia)$, this is equivalent to marginalising out DM_{ba}, effectively projecting the joint distribution in Figure 6.4 onto the DM_{ia} axis, and integrating the marginal distribution using DM'_{ia} obtained from $N(ia)'$, e.g.

$$\begin{aligned}\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) &= \int_{t=\mathrm{DM}'_{ia}}^{\infty} g_{ia}(t)dt \\ &= 1 - \Phi\Big(\frac{\mathrm{DM}'_{ia}\,\mu_w - \mu_z}{\sqrt{v_z - 2c\,\mathrm{DM}'_{ia} + v_w\,\mathrm{DM}'^2_{ia}}}\Big).\end{aligned} \tag{6.12}$$

This is illustrated by the shaded areas in Figure 6.4.

The second requirement for correctly mining the split is to ensure the $b \rightarrow a$ arc is not retained, by meeting conditions (3.4)–(3.6). The method above extends to calculating requirement (3.6), $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba} + \mathrm{RTB})$, equivalent in Figure 6.4 to shifting the $\mathrm{DM}_{ia} = \mathrm{DM}_{ba}$ line to the right by RTB, making it harder to concentrate the distribution to the right of the line by reducing its variance with increasing n. Let $R = \mathrm{RTB}$,

$$\begin{aligned}&\mathrm{DM}_{ia} = \mathrm{DM}_{ba} + R = \frac{m - 2N(ia)'}{m+1} + \frac{R(m+1)}{m+1} \\ &\Rightarrow N(ia)'(m+1) = \big(N(ia)'+1\big)\big(m - 2N(ia)' + R(m+1)\big) \\ &\Rightarrow 2N(ia)'^2 + \big(3n - R(m+1)\big)N(ia)' - \big(m + R(m+1)\big) = 0.\end{aligned}$$

From which $N(ia)'$ is obtained to calculate the probability from (6.11) as before.

The probabilities of meeting the DT and PO requirements (3.4) are obtained from

$$\begin{aligned}&\gamma_n\big(\,\mathrm{DM}_{ba} < \mathrm{DT}\,\big) = \int_{t=-\infty}^{\mathrm{DT}} g_{ba}(t)dt, \\ &\gamma_n\big(N(ba) < \mathrm{PO}\,\big) = \gamma_n\big(N(ib) < \mathrm{PO}\,\big) = \sum_{N(ib)=1}^{\mathrm{PO}-1} Q_n\big(N(ib)\big),\end{aligned} \tag{6.13}$$

The final requirement is $N(ia) > \text{PO}$ and $N(ib) > \text{PO}$:

$$\gamma_n\big(N(ia) > \text{PO} \wedge N(ib) > \text{PO}\big) = \sum_{N(ia)=\text{PO}+1}^{n} \sum_{N(ib)=\text{PO}+1}^{n} Q_n\big(N(ia), N(ib)\big),$$

i.e. for n traces drawn from a multinomial distribution with three outcomes, where a trace contains either ia or ib (not both), or neither, more than PO traces contain ia, and more than PO contain ib.

Combining the previous requirements,

$$\begin{aligned} P_{HM,n}\big(i \rightarrow_n (a \parallel b)\big) &= \gamma_n(\text{DM}_{ie} > \text{DM}_{fe}) \\ &\times \gamma_n\big(N(fe) < \text{PO} \vee \text{DM}_{fe} < \text{DT} \vee |\,\text{DM}_{ie} - \text{DM}_{fe}\,| > \text{RTB}\big) \\ &\times \gamma_n\big(N(ia) > \text{PO} \wedge N(ib) > \text{PO}\big), \end{aligned} \tag{6.14}$$

where $e, f \in \{a, b\}, e \neq f$ such that $\mathbb{E}_{Q_n(ef,fe)}[\text{DM}_{ef}] > 0$, i.e. obtain the probability using the 'more difficult' requirement, the Dependency Measure which is likely to be positive.

Equation (6.14) assumes the requirements to be independent, whereas in reality there will be some positive correlation (e.g. as n increases, the probabilities of meeting the requirements will all increase). The calculated probability of correct mining will therefore be underestimated. In practice we identify the number of traces such that each requirement is met with probability at least $1 - \epsilon$ (for small $0 < \epsilon \ll 1$), and therefore $P_{HM,n} \geq 1 - \epsilon$.

Experimental Evaluation of AND2 Method

Table 6.1 (full version in Appendix Table D.1) records the predicted numbers of traces using the described methods, for $\gamma_n(\text{DM}_{ia} > \text{DM}_{ba}) > 0.95$, for $\pi(ia), \pi(ib) \in [0, 1]$ in intervals of 0.05. The table also shows the number of traces for both $N(ia)$ and $N(ib)$ to exceed PO, and for a single DM to exceed DT. PO is the the determining factor in mining a split (as AND2 rather than XOR) except for highly imbalanced splits.

This is seen in Figure 6.5(a), which shows the number of traces to meet the combined

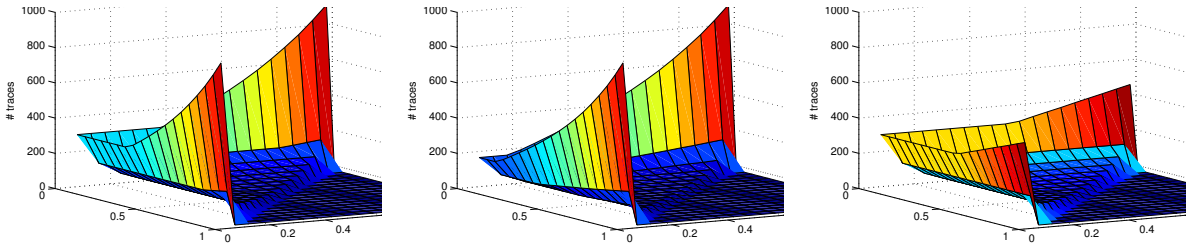
$\pi(ib)$	$\pi(ia) = 0.05, 0.1, \ldots, 1.0$										
0.05	109	46	28	20	15	12	9	8	6	...	1
0.1	133	53	31	21	16	12	10	8	7		
...											
0.55	299	99	53	34	24	18	14	11	8		
	316	104	55	35	25	18	14	11			
	332	108	57	36	25	19	15				
	348	113	59	38	26	20					
	364	117	61	39	27						
	379	121	63	40							
	395	126	65								
	410	130									
0.95	**425**										
PO = 10	311	154	102	76	60	49	42	36	32	28...	
PO = 5	181	89	59	44	34	28	24	21	18	16...	
DT = 0.9	306	151	99	74	58	48	40	35	30	27...	

Table 6.1: Top: Predicted Number of Traces for AND2 $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) \geq 0.95$, Middle: $\gamma_n\big(N(ia) > \mathrm{PO} \wedge N(ib) > \mathrm{PO}\big) \geq 0.95$, Bottom: $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DT}) \geq 0.95$.

requirements (6.14). At the peaks, the extra arc remains until the RTB requirement is met. Reducing PO (Figure 6.5(b)) enables discovery in fewer traces except at these peaks, which then extend to more cases where one probability is small. Reducing RTB (Figure 6.5(c)) reduces the number of traces for $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba} + \mathrm{RTB}) > 0.95$ at these peaks. Simulation shows that these predictions are slight overestimates (Appendix Table D.2), so represent a safe lower bound for mining.

6.2.5 3-Way Parallel Splits (AND3)

For a 3-way parallel split ('AND3') from activity i to paths beginning a, b or c, there are now 6 possible sequences of activities, $iabc, iacb, ibac, ibca, icab, icba$, and 3 pairs of Dependency Measure requirements: (i) $\mathrm{DM}_{ia} > \mathrm{DM}_{ba}$ and $\mathrm{DM}_{ib} > \mathrm{DM}_{ab}$, (ii) $\mathrm{DM}_{ia} > \mathrm{DM}_{ca}$ and $\mathrm{DM}_{ic} > \mathrm{DM}_{ac}$, and (iii) $\mathrm{DM}_{ib} > \mathrm{DM}_{cb}$ and $\mathrm{DM}_{ic} > \mathrm{DM}_{bc}$. As for AND2, in each of these pairs, one DM is the negation of the other, so only three requirements need be satisfied, plus the requirements from Equations (3.4)–(3.6) to ensure the extra arcs $b \to a$ or $a \to b$, etc. are not created, and $N(ia), N(ib), N(ic) > \mathrm{PO}$.



(a) (b) (c)

Figure 6.5: Predicted Number of Traces for $P_{HM,n}\big(i \rightarrow (a \parallel b)\big) \geq 0.95$ (Equation 6.14), plotted against $\pi(ia), \pi(ib) \in [0, 1]$. (a) PO $= 10, \mathrm{DT} = 0.9, \mathrm{RTB} = 0.05$, (b) reducing PO makes Discovery easier except at the Peaks, where RTB determines Discovery, (c) reducing RTB reduces the Height of the dominating Peaks.

We continue to consider $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$. Now $N(ba)$ is the sum of $N(ibac)$ and $N(icba)$ and likewise, $N(ab) = N(iabc) + N(icab)$. The counts $N(ia)$, $N(ibac)$, $N(ibca)$, $N(icab)$, $N(icba)$ can be considered as the 5 outcomes of a Multinomial distribution with probabilities $\pi(ia), \pi(iabc)$, etc. and in total m traces which pass through the split. We redefine $Q_n\big(N(ia)\big)$ as the probability of ia occurring $N(ia)$ times in event log $\mathcal{W}$ under this Multinomial distribution, $Q_n\big(N(ibac)|N(ia)\big)$ for the conditional probability of $ibac$ occurring $N(iabc)$ times given $N(ia)$ occurrences of ia, and $Q_n\big(N(ia), N(ibac)\big)$ as the joint probability of $N(ia)$ and $N(iabc)$. We write $c\big(N(ia), N(iabc)\big)$ for the correlation between $N(ia)$ and $N(iabc)$, etc.

Lemma 2. *Correlation $c\big(N(ia), N(ibac)\big)$ between $N(ia)$ and $N(ibac)$ is negative,*

$$c\big(N(ia), N(ibac)\big) = -\sqrt{\frac{\pi(ia) \cdot \pi(ibac)}{(1 - \pi(ia))(1 - \pi(ibac))}} < 0,$$

Proof. $N(ia)$ and $N(ibac)$ are two outcomes of a Multinomial distribution, similarly $c\big(N(ia), N(ibca)\big)$, $c\big(N(ia), N(icab)\big)$, $c\big(N(ia), N(icba)\big)$. □

Lemma 3. *Correlation $c\big(N(ia), N(iabc)\big)$ between $N(ia)$ and $N(iabc)$ is positive,*

$$c\big(N(ia), N(iabc)\big) = \frac{\pi(iabc)}{\pi(ia)} > 0.$$

Proof. ia is followed by b with probability $\pi(b|ia)$, otherwise by c. Otherwise it is not

representable in our framework. The correlation is the conditional probability of the trace containing iab given that it contains ia. □

We have the Dependency Measures

$$\begin{aligned}
\mathrm{DM}_{ia} &= \tau\big(N(ia), 0\big), \\
\mathrm{DM}_{ba} &= \tau\big(N(ba), N(ab)\big) = \tau\big(N(ibac) + N(icba), N(iabc) + N(icab)\big).
\end{aligned}$$

DM_{ba} is conditionally dependent on DM_{ia} due to the correlations between string counts.

Correlation $c\big(N(ia), N(iabc)\big)$ gives the strength of the linear relation between $N(ia)$ and $N(iabc)$. Therefore given observed $N(ia) = N(ia)'$ we obtain a conditional expected value for $N(iabc)$. Let δ_{ia} be the difference between the expected $N(ia)$ and the observed value $N(ia)'$,

$$\delta_{ia} = N(ia)' - \mathbb{E}_{Q_n(N(ia))}[N(ia)].$$

Then the expected value of $N(iabc)$ conditional on $N(ia) = N(ia)'$ is

$$\begin{aligned}
\mathbb{E}_{Q_n(N(iabc)|N(ia)')}[N(iabc)] &= \mathbb{E}_{Q_n(N(iabc))}[N(iabc)] + c\big(N(ia), N(iabc)\big)\delta_{ia} \\
&= \mathbb{E}_{Q_n(N(iabc))}[N(iabc)] + c\big(N(ia), N(iabc)\big)\big(N(ia)' - \mathbb{E}_{Q_n(N(ia))}[N(ia)]\big),
\end{aligned} \tag{6.15}$$

and we obtain conditional expected values of $N(ba)$ and $N(ab)$,

$$F(ab) = \mathbb{E}_{Q_n(N(ab)|N(ia)')}[N(iab)] = \mathbb{E}_{Q_n(N(iabc)|N(ia)')}[N(iabc)] + \mathbb{E}_{Q_n(N(icab)|N(ia)')}[N(icab)], \tag{6.16}$$

$$F(ba) = \mathbb{E}_{Q_n(N(ba)|N(ia)')}[N(iba)] = \mathbb{E}_{Q_n(N(ibac)|N(ia)')}[N(ibac)] + \mathbb{E}_{Q_n(N(icba)|N(ia)')}[N(icba)]. \tag{6.17}$$

Writing the expected value of DM_{ba} given DM_{ia} calculated from $N(ia)'$, the observed

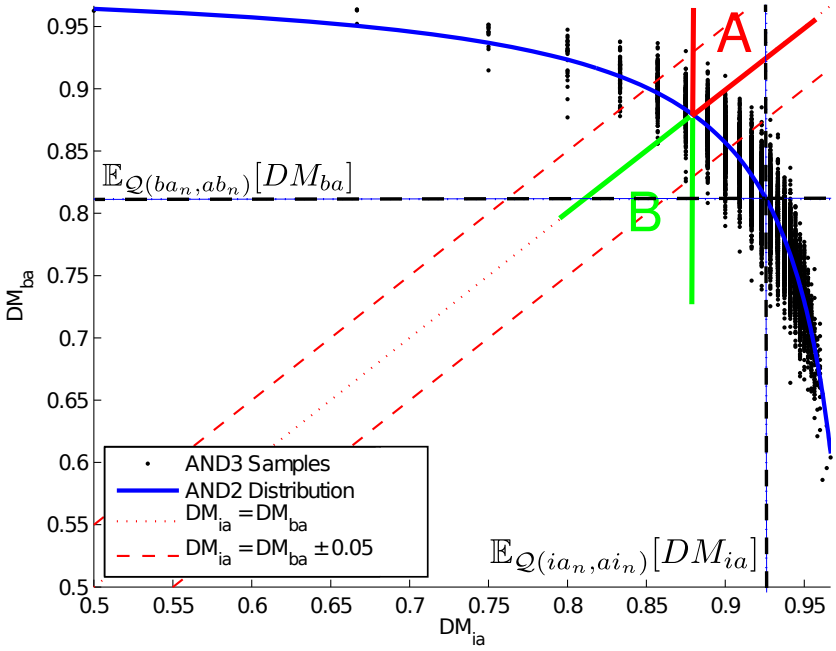


Figure 6.6: Example 'AND3' distribution for $\pi(ia) = 0.05$, $\pi(ib) = 0.9$, $\pi(ic) = 0.05$, $n = 250$ Traces, $\pi(b|ia) \propto \pi(ib)$, i.e. $\pi(b|ia) = \frac{\pi(ia)}{1-\pi(ia)}$, etc. For A and B see Text.

value of $N(ia)$,

$$\begin{aligned}
&\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}] = \mathbb{E}_{Q_n(N(ba),N(ab)|N(ia)')}[\mathrm{DM}_{ba}] \\
&= \sum_{N(ba)=0}^{n} \sum_{N(ab)=0}^{n} Q_n\big(N(ba), N(ab)|N(ia)'\big)\tau\big(N(ba), N(ab)\big) \\
&= \sum_{N(ba)=0}^{n} \sum_{N(ab)=0}^{n} Q_n\big(N(ba)|N(ia)'\big).Q_n\big(N(ab)|N(ba), N(ia)'\big)\tau\big(N(ba), N(ab)\big).
\end{aligned}$$

which, as in Section 6.1.1, we can rewrite

$$\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}] = \tau\big(F(ba), F(ab)\big),$$

again confirmed by simulation and numerical integration. This gives a functional dependency between DM_{ia} and the conditional expected value of DM_{ba}, illustrated by the curve in Figure 6.6. We next show the dependency to be negative.

Proposition 8. *For all relevant DM 'requirements',* $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ *is nega-*

tively related to DM_{ia}*, i.e. as* DM_{ia} *increases,* $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ *decreases.*

Proof. The proof is similar to that for Proposition 7 and given in full in Appendix A.2. □

We therefore approximate $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$ using the 'AND2 method' described for 2-way parallel splits, Section 6.2.4, Equations (6.10)–(6.12). Effectively, the joint distribution of DM_{ia} and DM_{ba} is projected onto the line given by plotting DM_{ia} against $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ for all values of $N(ia) \in [0, m]$, ignoring variation in DM_{ba}. Figure 6.6 shows an example distribution. For the part of the distribution labelled A, DM_{ba} is less than DM_{ia} but will be counted as larger. This will be approximately equal to part B for which the reverse will be the case.

Using the following constants defined in the proof (Appendix Section A.2.1),

$$\begin{aligned}
D_1 &= m\big[\pi(ibac) + \pi(icba) - \pi(iabc) - \pi(icab) - \pi(ia)\big(c\big(N(ia), N(ibac)\big) + \\
&\qquad c\big(N(ia), N(icba)\big) - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big)\big)\big], \\
D_2 &= c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big), \\
D_3 &= m\big[\pi(ibac) + \pi(icba) + \pi(iabc) + \pi(icab) - \pi(ia)\big(c\big(N(ia), N(ibac)\big) + \\
&\qquad c\big(N(ia), N(icba)\big) + c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big)\big)\big] + 1, \text{ and} \\
D_4 &= c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) + c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big).
\end{aligned} \tag{6.18}$$

we can write the expected value of DM_{ba} as follows,

$$\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}] = \frac{D_1 + D_2 N(ia)}{D_3 + D_4 N(ia)},$$

and rework (6.10) to obtain $N(ia)'$ for which $\mathrm{DM}_{ia} = \mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$:

$$\begin{aligned}
\frac{N(ia)'}{N(ia)'+1} &= \frac{D_1 + D_2 N(ia)'}{D_3 + D_4 N(ia)'} \\
&\Rightarrow N(ia)'\big(D_3 + D_4 N(ia)'\big) = \big(N(ia)'+1\big)\big(D_1 + D_2 N(ia)'\big) \\
&\Rightarrow N(ia)'^2(D_4 - D_2) + N(ia)'(D_3 - D_1 - D_2) - D_1 = 0.
\end{aligned} \tag{6.19}$$

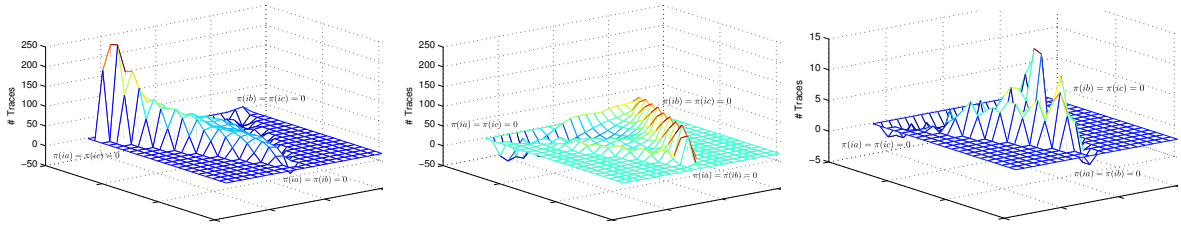


Figure 6.7: Difference between Predicted and Simulated Traces for 'Extreme' AND3 Probabilities. (a) Approximation using 'AND2 Method', (b) DMs Approximated with Gaussians, (c) Minimum of (a) and (b). Positive Difference indicates Predictions are Underestimates.

$\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$ is obtained using $N(ia)'$ obtained from (6.19), in (6.11).

Equation (6.19) can be extended for $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba} + \mathrm{RTB})$. Let $R = \mathrm{RTB}$, then

$$\begin{aligned}
\frac{N(ia)'}{N(ia)'+1} = \frac{D_1 + D_2 N(ia)'}{D_3 + D_4 N(ia)'} + R &= \frac{D_1 + D_2 N(ia)' + R\big(D_3 + D_4 N(ia)'\big)}{D_3 + D_4 N(ia)'} \\
\Rightarrow N(ia)'\big(D_3 + D_4 N(ia)'\big) &= \big(N(ia)'+1\big)\big((D_1 + D_3 R) + (D_2 + D_4 R) N(ia)'\big) \\
\Rightarrow N(ia)'^2\big(D_4(1-R) - D_2\big) + N(ia)'\big(D_3(1-R) - D_1 - D_2 - D_4 R\big) & \\
- (D_1 + D_3 R) = 0. &
\end{aligned}$$

The three requirements to satisfy to mine AND3 correctly are $\mathrm{DM}_{if} > \mathrm{DM}_{ef}$, such that $\mathbb{E}_{Q_n(ef,fe)}[\mathrm{DM}_{ef}] > 0$. $e, f \in \{a, b, c\}, e \neq f$. Without loss of generality, assume these to be $\mathrm{DM}_{ia} > \mathrm{DM}_{ba}, \mathrm{DM}_{ia} > \mathrm{DM}_{ca}, \mathrm{DM}_{ib} > \mathrm{DM}_{cb}$. Then we approximate by multiplying the probabilities,

$$\begin{aligned}
&P_{HM,n}\big(i \rightarrow_n (a \parallel b \parallel c)\big) \geq \\
&\quad \gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) \times \gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ca}) \times \gamma_n(\mathrm{DM}_{ib} > \mathrm{DM}_{cb}) \\
&\quad \times \gamma_n\Big(N(ba) < \mathrm{PO} \vee \mathrm{DM}_{ba} < \mathrm{DT} \vee |\,\mathrm{DM}_{ia} - \mathrm{DM}_{ba}\,| > \mathrm{RTB}\Big) \times \ldots \\
&\quad \times \gamma_n\Big(N(ia) > \mathrm{PO} \wedge N(ib) > \mathrm{PO} \wedge N(ic) > \mathrm{PO}\Big). \qquad (6.20)
\end{aligned}$$

As for AND2, this assumes the probabilities of the multiple requirements to be independent, which will tend to underestimate the probability.

Experimental Evaluation of AND3 Method

Using (6.20) we predicted the traces needed for 95% probability of correctly mining AND3 splits, for all combinations of the probability of the first activity (a, b or c) after the split, i.e. $\pi(ia) + \pi(ib) + \pi(ic) \in [0, 1]$, in intervals of 0.05. Rather than attempt to record and visualise predictions in the six dimensions produced by also allowing all possible variations in the probabilities of the second activity after the split, we made predictions using three different assumptions for the behaviour of the probabilities of this second activity, i.e. the conditional probabilities $\pi(b|ia)$, $\pi(c|ia)$, $\pi(a|ib)$, $\pi(c|ib)$, $\pi(a|ic)$ and $\pi(b|ic)$:

1. 'Proportional': We assume the probability of activity a occurring as second activity after the split, after b or c, to be proportional to its probability of occurring first. So if a has low probability of being the first activity after i, then it also has proportionally low probability of following b or c. Thus $\pi(a|ib) = \frac{\pi(ia)}{\pi(ia)+\pi(ic)}$, $\pi(c|ib) = \frac{\pi(ic)}{\pi(ia)+\pi(ic)}$, and so on.
2. 'Even': Here we simply assume that after the first activity following the split, the remaining two occur with equal probability as the second activity in the path, i.e. $\pi(b|ia) = \pi(c|ia) = 0.5$, $\pi(a|ib) = \pi(c|ib) = 0.5$, $\pi(a|ic) = \pi(b|ic) = 0.5$.
3. 'Extreme': Here we assume that one of the two remaining activities is significantly more likely than the other, i.e. $\pi(b|ia) = \pi(a|ib) = \pi(a|ic) = 0.05$, and $\pi(c|ia) = \pi(c|ib) = \pi(b|ic) = 0.95$.

Appendix Tables D.4, D.5 and D.6 record the difference between predicted and simulated traces to meet the Dependency Measure requirements in the three cases.

For splits with 'proportional' second probabilities, the predictions slightly overestimate the numbers of traces, so represent a safe lower bound on the number of traces to use for mining. Similarly for 'even' second probabilities, except for slight underestimation where one of $\pi(ia), \pi(ib), \pi(ic)$ is relatively small. However, with 'extreme' probabilities, $\pi(ba)$ or $\pi(ab)$ is often small (near 0) or large (near 1). The method significantly overestimates the probabilities in many cases (Figure 6.7a). In most of these cases, the

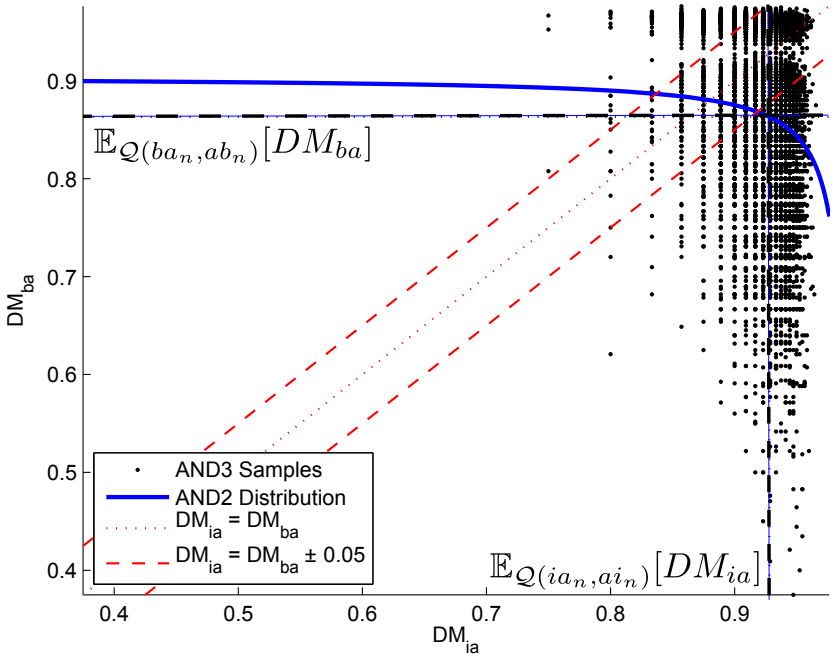


Figure 6.8: Dual-Peak Joint Distribution, $\pi(ia) = 0.05$, $\pi(ib) = 0.9$, $\pi(ci) = 0.05$, $n = 250$ Traces, 'Extreme' Probabilities $\pi(b|ia) = 0.05$, $\pi(c|ia) = 0.95$.

Gaussian approximations to the Binomial random variables are poor, e.g. one or more of $n\pi(ab), n(1-\pi(ab)) \leq 5$. The DM_{ba} distribution then has two peaks, seen in the simulated joint distribution in Figure 6.8, which causes the approximation of the distribution by the line mapping DM_{ia} to $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ to incur too much error.

We next introduce a simple approximation method, which is successful where approximation using the 'AND2 method' described in Section 6.2.5 fails.

Approximating Dependency Measures with Gaussians

Rewriting the PDF for the Dependency Measure (6.4) suggests approximation by a Gaussian. Denoting by Z the the normalisation factor,

$$\begin{aligned} g_{ba}(t) &= Z^{-1}\phi\Big(\frac{t\mu_w - \mu_z}{\sqrt{v_z - 2tc + t^2 v_w}}\Big) = Z^{-1}\phi\Big(\frac{t - \frac{\mu_z}{\mu_w}}{\frac{1}{\mu_w}\sqrt{v_z - 2tc + t^2 v_w}}\Big), \\ &= Z^{-1}\phi\Big(\frac{t - \mathbb{E}_{Q_n(N(ba),N(ab))}[\mathrm{DM}_{ba}]}{\frac{1}{\mu_w}\sqrt{v_z - 2tc + t^2 v_w}}\Big), \end{aligned} \tag{6.21}$$

implying the Gaussian approximation to the Dependency Measure

$$g'_{ba}(t) \sim \mathcal{N}(\mu, \sigma^2), \text{ where } \mu = \frac{\mu_z}{\mu_w} = \mathbb{E}_{Q_n(ba,ab)}[\mathrm{DM}_{ba}], \tag{6.22}$$
$$\sigma = \frac{1}{\mu_w}\sqrt{v_z - 2c\mathbb{E}_{Q_n(ba,ab)}[\mathrm{DM}_{ba}] + v_w(\mathbb{E}_{Q_n(ba,ab)}[\mathrm{DM}_{ba}])^2}.$$

Since the skew in the original distribution is ignored, the behaviour will be different as the DMs are 'separated' as n increases. This turns out to be a good approximation in those cases when approximation with the 'AND2 method' underestimates.

To estimate $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$ we approximate each DM in this way and also assume they are independent. This assumption is reasonable because when the Binomial distributions of $N(ba)$ or $N(ab)$ are not approximated well by Gaussians, the probabilities $\pi(ba), \pi(ab)$ must be close to 0 or 1. In either case, the correlations with $N(ia)$ will be small, hence low correlation between DM_{ia} and DM_{ba}. The joint distribution is translated by the means of both Gaussians (the DM values) and scaled by the standard deviations. $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$ can then be calculated from a Standard Normal distribution, using the distance from the origin to the transformed $\mathrm{DM}_{ia} = \mathrm{DM}_{ba}$ line.

We give further details of this method in Appendix C.

This method underestimates by up to 40% the number of traces needed for mining in cases where the predictions by the 'AND2 method' are good, but overestimates in approximately the cases where that method underestimates. Figure 6.7b shows the differences between the predicted and simulated numbers of traces. Figure 6.7c shows the maximum difference between simulations and predictions (i.e. errors), for both methods. In most cases, one of the predictions, chosen as follows, gives an adequate approximation:

- The method in Section 6.2.5 should be used when the Binomially distributed counts ($N(ia), N(iabc)$, etc.) can with acceptable accuracy be approximated by Gaussians.
- The approximation in this section should be used where this approximation is not accurate enough for one or more probabilities.

Finally we note that the methods for 3-way parallel splits also apply to 2-way splits

where either other parts of the model are in parallel with the split, or the paths following the split contain more than one activity. In both cases there are more possible strings, e.g. $\pi(iaa') > 0$ where $a' \in \Sigma \setminus \{i, a, b\}$.

Parallel joins, i.e. paths ending with a, b and c, joining to o, are treated in the same way, e.g. satisfying DM requirements $\mathrm{DM}_{ao} > \mathrm{DM}_{ab}$ and $\mathrm{DM}_{ao} > \mathrm{DM}_{ac}$, and so on.

6.2.6 Splits and Joins with More than 3 Paths

Where more than 3 paths follow the split, or there are other parts of the model in parallel with the split, the 'AND3 method' (Section 6.2.5) can be used if all of the sub-string probabilities and correlations can be identified. These are included in the factors (6.18), then (6.19) is applied. This will underestimate the probabilities, since it assumes the DMs to be negatively correlated, and hence overestimate the number of traces needed.

As the number of parallel paths increases, the correlations between the probabilities of different sub-strings will tend to reduce (since they are drawn from a Multinomial distribution with increasing number of outcomes), so the DMs also become less correlated. Therefore the method of approximating the DMs by Gaussians will become increasingly accurate and more practical.

6.2.7 Other Structures

In structured process models, in addition to sequences, splits and joins, we also find 'extra parallelism' and complex splits and joins. If after activity i there are 3 parallel paths, e.g. $a \to a'$, $b \to b' \to b''$, and $c \to c'$ (Figure 6.9), two concerns are introduced. Firstly that the parallel split is mined correctly, and secondly that no extra arcs such as $b \to a'$ are introduced. For the split, traces such as $ibb'a$ are possible, so DM_{ia} must not only be greater than DM_{ba} and DM_{ca}, but also greater than $\mathrm{DM}_{b'a}, \mathrm{DM}_{c'a}$, etc.:

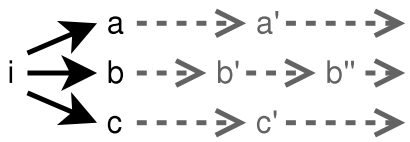


Figure 6.9: Illustration of Extra Activities in Parallel paths.

$$\mathrm{DM}_{ia} > \mathrm{DM}_{ja}, \forall\, j \in \{b, b', b'', c, c'\},$$
$$\mathrm{DM}_{ib} > \mathrm{DM}_{kb}, \forall\, k \in \{a, a', c, c\}, \text{ and}$$
$$\mathrm{DM}_{ic} > \mathrm{DM}_{la}, \forall\, l \in \{a, a', b, b', b''\}.$$

These are dealt with as in Section 6.2.6, as a split to more than three parallel paths.

We also require activities in different paths to be mined as parallel (e.g. $(a' \parallel b')$). This is the same as ensuring that sequences $(a \rightarrow a')$, etc. do not become splits or joins, by the creation of extra arcs. This is equivalent to local 'noise' affecting the correct mining of the sequences; we discuss it in the context of mining from noisy logs, in Section 8.2.

'Complex' splits combining XOR and parallel splits can be handled by combining the methods in the previous sections.

6.3 Experimental Evaluation Without Noise

We used the methods described in the previous sections to predict the number of traces needed for correct mining of the example process, shown again in Figure 6.10 with the structures highlighted. Let this be process model $\mathcal{M}$, then (as Section 5.3.3, Chapter 5),

$$P_{HM,n}(\mathcal{M}) = P_{HM,n}(A) \times P_{HM,n}(B|A) \times P_{HM,n}(C|B) \times P_{HM,n}(D|C)$$
$$\times P_{HM,n}(E|D) \times P_{HM,n}(F|B \wedge E).$$

$P_{HM,n}(B|A)$ is the probability of correct mining of structure B given A correctly mined.

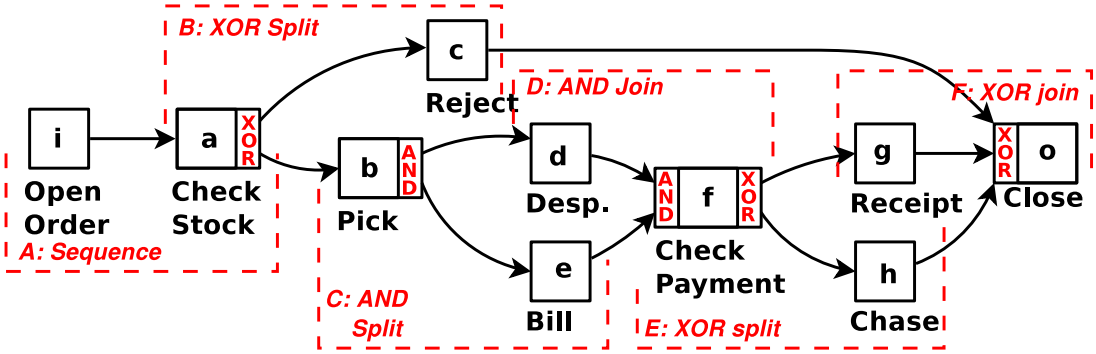


Figure 6.10: Process Structures in the Example Process $\mathcal{M}$.

κ	defaults	RTB		PO		DT	
		0.01	0.1	5	1	0.5	0.95
0	84	84	84	**49**	**45**	84	84

Table 6.2: Predicted Number of Traces needed for correct mining from Noise-Free Logs.

We used automaton A_0 (Figure 2.5) to predict the numbers of traces needed for mining using Heuristics Miner from noise-free logs, for various values of HM's parameters. These predictions are shown in Table 6.2. The only parameter that affects mining this model is PO, indicating that seeing enough traces to decide the split is parallel, not XOR, is the determining requirement.

To verify these predictions experimentally, the automaton was randomly walked to produce logs of traces in the MXML format [167] (10 of each number of traces), i.e. samples from P_M, the distribution represented by the true business process. A large 'ground truth' log of $10,000$ traces was also produced. The Heuristics Miner implementation and conversion plugins in ProM 5.2 [170] were used to mine process models from these logs and convert them to Petri nets, which were converted to probabilistic automata as for Alpha (Chapter 5). The example graph in Figure 6.11 illustrates that convergence is as predicted, measured by the average JSD distance (4.2) between P_M and $P_{M'}$, the distribution represented by the mined model, calculated using methods described by Cortes *et al.* [42]. Distances are averaged over the 10 models mined from each log size. The graph also shows the number of mined models of each size for which JSD was below 0.05.

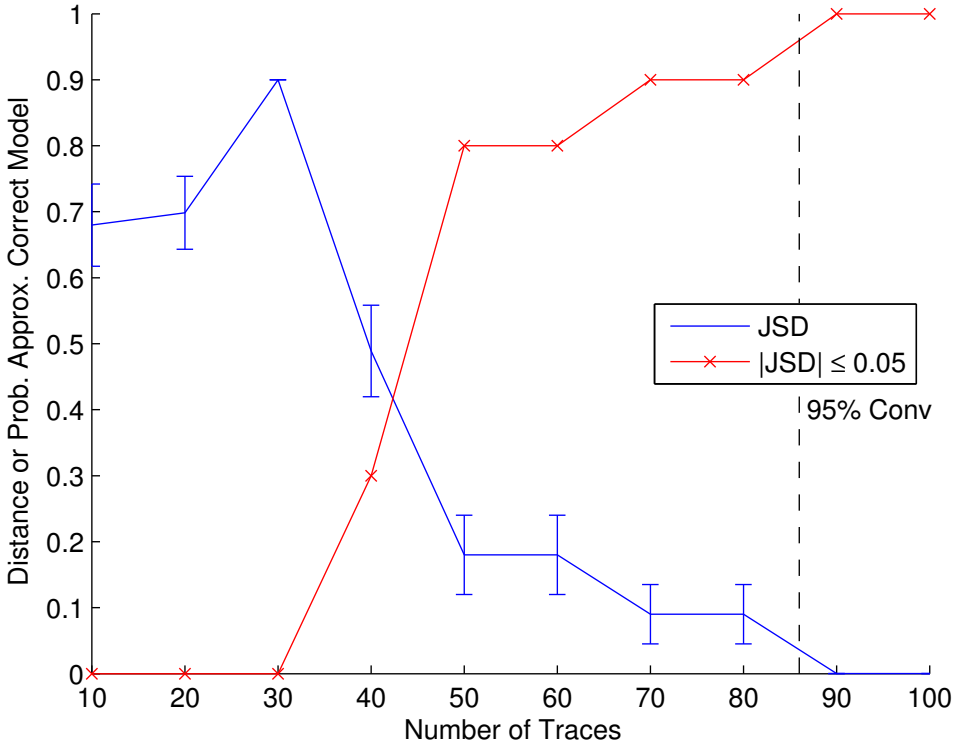


Figure 6.11: Convergence of Mining, from Noise-Free Logs: Average Approximate Model Correctness (JSD), and Probability of Approximately Correct Model ($|JSD| < 0.05$), plotted against Number of Traces.

6.4 Chapter Summary

In this chapter we applied our framework to the Heuristics Miner algorithm [194]. We discovered that although the algorithm is simple to describe and specify, this masks complex probabilistic behaviour. We took the theoretical analysis as far as was possible, then presented empirical results to confirm the validity of approximate methods for predicting the amount of data needed for correct mining.

The work in this chapter shows that the methods presented in the previous two chapters are of practical use in real-world situations, and provide a useful framework within which to consider process mining behaviour, even when full analytical analysis is not possible. In Chapter 8, we apply the analysis of Heuristics Miner to investigate 'noise' in process mining. First, in Chapter 7, we apply the framework to process mining in non-stationary environments.

CHAPTER 7

APPLICATION: PROCESS MINING IN NON-STATIONARY ENVIRONMENTS

In the previous chapters we introduced a probabilistic framework for considering process mining (Chapter 4) and applied it to analyses of the behaviour of the Alpha Algorithm [156] (Chapter 5) and Heuristics Miner [194] (Chapter 6). We now apply the framework and analyses to a practical application, process mining in non-stationary environments.

Process mining algorithms tend to assume the underlying process to be fixed, but this is unlikely to be the case in reality. The Process Mining Manifesto [149] lists one of the main challenges for process mining (Challenge C4) as detection of change in the underlying process, and analysis of the impact of the change on the process mining activity.

Businesses operate in real time, under pressures such as time, cost, competition and the need to continually customise their proposition to the market [114]. Business processes play a key rôle in managing the business, allowing it to react to changes in the market, financial situation or legislature; and detect and respond to problems in a timely manner. Therefore processes need to be adaptable, to allow rapid response to the changing environment [118]. Process mining, supporting these processes, cannot assume stationarity.

We first consider how our framework supports 'Real-Time Business Process Mining', and apply this idea to detecting process change. We then show how using our framework we can recover the sequence of changed process models over time from a business process.

Much of the material in this chapter was originally presented in [186, 188].

7.1 Real Time Business Process Mining (RTBPM)

One approach to mining from a non-stationary process is to use only a subset of an event log, within which the process is known to be stationary. To do so, we need a method to determine what is the minimum number of traces to use for mining, to be confident in mining the correct process model. In Chapter 4 we introduced one such method, and in Chapters 5 and 6 applied it to the Alpha and Heuristics Miner algorithms. Knowing this minimum number of traces is also beneficial if process data is expensive or time-consuming to collect, or the mining algorithm is computationally expensive.

We consider 'real-time' process mining, by asking how accurate an algorithm can be on a limited data set. However, the term 'real-time' is used with varying rigour.

1. Informally, but subjectively, systems which appear to process information 'fast', or update it as it is received. For example, streaming video in acceptable quality.
2. Formally, real-time systems 'must react within precise time constraints to events in the environment' [30]. They impose timing bounds: times or events before which data will not be available or tasks cannot start, and after which data will not be useful, or tasks must have completed [94, 98].

Predictability and results guaranteed within a specified timeframe, rather than speed, is key, 'Hard' real-time systems consider late results as wrong, perhaps catastrophically. 'Soft' real-time allows some flexibility. Nevertheless, understanding timing is critical [115].

Goals of process mining suggest the restrictions of 'soft' real-time are relevant. Businesses are interested in identifying and understanding differences between the 'believed' and actual process, which may necessitate decisions to respond to these findings. For process mining to support the 'real-time enterprise' [114] a model should be mined that is not only correct, but also produced within a guaranteed time. Processing time can be linked to both the complexity of the algorithm, and the amount of data needing to be processed, so we can impose two main constraints on the process mining activity:

Constraint 1. Accuracy of the Mined Model. *Process mining algorithm $\mathcal{L}$, mining from*

event log $\mathcal{W}_n$ of n traces produced by an underlying ground truth process $\mathcal{M}$, should with probability $1-\delta$ return model $\mathcal{M}' = \mathcal{L}(\mathcal{W}_n)$ such that $d(P_M, P_{M'}) \leq \epsilon$, for $0 < \delta, \epsilon \ll 1$ and some notion of distance $d(P_M, P_{M'})$ between the true (P_M) and inferred ($P_{M'}$) distributions over traces. This is equivalent to requiring

$$P_{\alpha,n}(\mathcal{M}) \geq (1-\delta), \qquad P_{HM,n}(\mathcal{M}) \geq (1-\delta),$$

for probability $P_{\alpha,n}(\mathcal{M})$ of correct mining of model $\mathcal{M}$ by the Alpha algorithm, or $P_{HM,n}(\mathcal{M})$ of correct mining using Heuristics Miner (Chapters 5, 6).

Constraint 2. Efficiency. *Assume that mining with algorithm $\mathcal{L}$, from event log $\mathcal{W}_n$ of n traces takes $s(\mathcal{L}, n)$ time steps, and $\mathcal{L}$ has fixed overhead of S steps and takes c steps to process a trace, then we can impose a real-time constraint η steps within which a result is required, such that*

$$s(\mathcal{L}, n) = S + nc \leq \eta. \tag{7.1}$$

Constraint (2) reflects the time within which a decision must be made, or a problem detected, typically expressed by the business in terms of time, such as number of seconds within which a result is required. Rearranging (7.1) gives an upper bound on the number of traces which can be used for mining, above which the time constraint will be violated: $n \leq \frac{(\eta - S)}{c}$ traces. Clearly $n > 0$, so $\eta > S$ is a lower bound on the time needed for mining.

From the previous chapters we know that some minimal number of traces n_{min} will be needed, such that the model accuracy constraint (1) is only satisfied when $n > n_{min}$. It is in our interest to understand this lower bound on traces, to be able to set realistic expectations for constraint (2), the time allowable for process mining. Our framework enables this understanding. We next turn to applying this to the detection of change in non-stationary processes, and the recovery of the sequence of changed models over time.

7.2 Process Mining in Non-Stationary Environments

In this section we consider process mining in non-stationary environments in an online manner. Business process research has discussed the need for flexibility and allowing for, and timely detection of, process change [118, 135, 183]. Questions of how much data is needed and how to identify when the process has changed, have been less investigated. In [3, 26] statistical tests on features in log files are used to identify where in a log file the process changed. Here we use our framework to propose a principled approach to efficiently mine and detect change to both model probabilities and structures, recovering the sequence of changed process models. The model changes we can detect are of more subtle nature than those detectable by re-estimation of standard process models, such as Petri nets.

7.2.1 Method for Model Estimation and Online Mining

We assume a real business process $\mathcal{M}$ generates traces into event log $\mathcal{W}$, and that we can model this process as a PDFA. These assumptions allow process mining with, for example, the Alpha [156] or Heuristics Miner [194] algorithms. For simplicity, and to avoid effects in the experimental results from factors other than change in the underlying process, in this chapter we also assume traces are generated without noise[1] and confine our discussion and experimentation to the Alpha algorithm.

The main idea behind the method is as follows. Since traces are generated randomly according to an unknown distribution, we use the behaviour of the mining algorithm and probabilities of traces to determine the minimum traces needed to be confident that the mining algorithm will create the correct model, and thus that if a different model is produced, the underlying model has changed, rather than being a feature of the sample.

To initially estimate $\mathcal{M}$, we use the most recent n_0 (a known over-estimate) traces from $\mathcal{W}$. We convert the Petri net mined by Alpha to a PDFA by labelling its reachability

[1] In this chapter we continue to refer informally to event logs without 'noise' as those recorded without error by an underlying process which is followed without error. We discuss 'noise' in Chapter 8.

graph with probability estimates derived from the frequencies of activities in the traces used for mining (Section 5.4.1). The distribution P_M that this automaton generates, is the estimate of the underlying model $\mathcal{M}$. We use the formulae developed in Chapter 5 for Alpha, to obtain n such that when mining with n traces we will with probability $1-\epsilon$ return the Petri net corresponding to the underlying model $\mathcal{M}$, for a desired confidence level $0 < \epsilon \ll 1$. Thus if mining produces a different model, we can have confidence $1-\epsilon$ that the underlying process has changed.

To detect change, we mine repeatedly from $\mathcal{W}_n$, the most recent n traces from event log $\mathcal{W}$, to obtain at each iteration a model $\mathcal{M}'$. Rather than use distances between distributions, for which it is not clear what distances are significantly significant, we use statistical and hypothesis tests, for detecting changes in the mined distribution or its PDFA representation (Sections 4.1.2 and 4.1.3).

We compare the distribution of traces in $\mathcal{W}_n$ with the ground truth estimate P_M using a Chi Square test [110, 136] (Section 4.1.2) to test the statistical significance of the differences between the observed ($N(x)$) and expected ($nP_M(x)$) frequencies of traces. We interpret the Chi^2 test p-value as indicating that, with probability $1-p$, the process has changed. We also use a hypothesis test (Section 4.1.3) to compare the probability under P_M of process traces occurring with the observed frequency, to determine whether the underlying distribution has changed. We assume the count $N(x)$ of trace x in $\mathcal{W}_n$ to be Binomially distributed. If the probability of the observed $N(x)$ is less than p under the hypothesis test, then with probability $1-p$ the distribution has changed.

While the process is unchanged, we assume that successive Petri nets mined by the Alpha algorithm will be the same, and therefore PDFA obtained from their reachability graphs (Section 2.3.1) will have the same state structure. Since Alpha does not assign probabilities to the mined model, we use frequencies of the traces in $\mathcal{W}$ to label arcs in the PDFA with probability estimates. Therefore as traces in $\mathcal{W}$ are randomly sampled from the underlying process $\mathcal{M}$, the probabilities in successive PDFA may vary even if the underlying distribution has not changed. We use a similar hypothesis test to compare

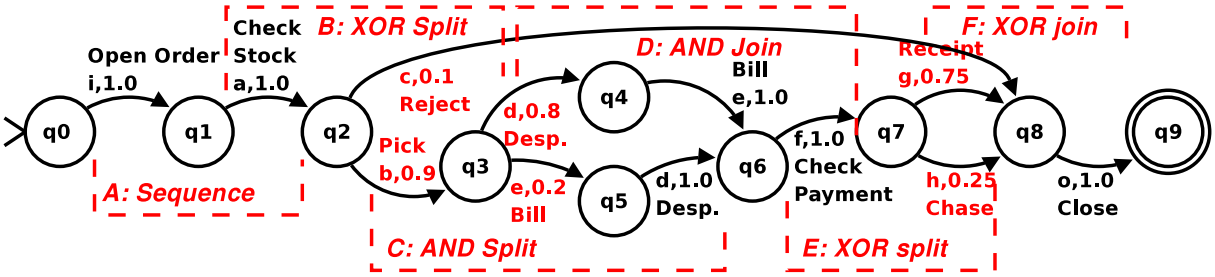


Figure 7.1: Order-fulfilment business process, as PDFA with structures highlighted.

the significance of differences between the probabilities assigned to equivalent arcs in successive models.

After detecting change, we wait for n traces, then re-estimate $\mathcal{M}$ and n.

7.2.2 Evaluation of Methods to Detect Change

Applying the analysis described in Chapter 5 to the running example as the initial underlying model (Figure 7.1), 45 traces are needed for 99% confidence in mining the correct Petri Net, but as probabilities vary this can increase to 500.

The results in Table 7.1 show the numbers of traces required to detect change using each of the methods described, for various changes introduced to the ground truth (Chi^2 p-value $= 0.01$, hypothesis test critical value $= 0.01$). In the first part of the table, probabilities in the XOR split (structure B) were varied from the ground truth values ($\delta_A(q_2, b, q_3) = 0.9, \delta_A(q_2, c, q_8) = 0.1$), through to $\delta_A(q_2, b, q_3) = 0.1, \delta_A(q_2, c, q_8) = 0.9$, in increments of 0.1. The 'pdiff' column shows the difference in $\delta_A(q_2, b, q_3)$ from the ground truth ($\delta_A(q_2, c, q_8) = 1 - \delta_A(q_2, b, q_3)$). In the centre part of the table, probabilities in the parallel split (structure C) were varied similarly. For this set of experiments, $\delta_A(q_2, b, q_3) = 0.9$, i.e. there was a high probability of traces passing through the parallel split. The lower part of the table shows the same experiments repeated with $\delta_A(q_2, b, q_3) = 0.1$, low probability of traces passing through the parallel split.

The 'KL' column shows the Kullback-Leibler divergence between the changed mined model and ground truth. The remaining columns show the number of traces needed for each statistical test to detect change: $\mathcal{X}^2$ for the Chi Square test, $h(s)$ for hypothesis

Change type	pdiff	KL	$\mathcal{X}^2$	h(s)	h(a)
XOR Split Probabilities	0.2	0.357	28	**27**	**27**
(Structure B)	0.3	0.430	22	**21**	**21**
	0.4	0.396	20	**19**	**19**
	0.5	1.190	15	15	**14**
	0.6	1.265	8	**7**	**7**
	0.7	∞	**6**	**6**	**6**
	0.8	2.099	9	**8**	**8**
Parallel Split Probabilities	0.1	0.150	58	58	**53**
(Structure C)	0.2	0.427	86	87	**84**
(with $\delta_A(q_2, b, q_3) = 0.9$)	0.3	0.724	24	24	**23**
	0.4	1.051	13	13	**8**
	0.5	0.813	10	9	**8**
	0.6	1.804	11	8	**6**
	0.7	2.036	16	16	**8**
Parallel Split Probabilities	0.2	0.033	207	-	**188**
(Structure C)	0.3	0.061	153	-	**139**
(with $\delta_A(q_2, b, q_3) = 0.1$)	0.4	0.071	**99**	-	**99**
	0.5	0.058	118	-	**100**
	0.6	0.035	73	-	**71**
	0.7	0.094	80	136	**75**

Table 7.1: Detection by several Tests, of Changes of various Types and Magnitudes. 'pdiff' indicates Change in Probability in the Structure, from the Ground Truth. $h(s)$ indicates Number of Traces to Detection by Hypothesis Test on Traces, $h(a)$ Hypothesis Test on Arc Probabilities, $\mathcal{X}^2$ Chi Square Test. KL shows Kullback-Leibler Divergence between Mined Model and Ground Truth. p-Value for the Chi Square test and Critical Value for the Hypothesis Tests were set to 0.01.

test on traces, and $h(a)$ for hypothesis test on arc probabilities. The entries in bold show which test was the first to detect the change[1].

Results show that change is detectable using the methods described and that more significant change is detected in fewer traces. Small variations (< 0.1) in the probabilities were not detectable, although the Kullback-Leibler Divergence was seen to increase. For the XOR split, the tests all identified the change after approximately the same number of traces. When variation of the parallel probabilities (structure C) was tested with low probability of the structure in the model ($\delta_A(q_2, b, q_3) = 0.1$) change was detected first by arc differences ($h(a)$). The string difference hypothesis test $h(s)$ failed to detect the

[1]These are representative results rather than averages over multiple runs.

changes. This is explained by the probability of traces passing through the AND structure being too low to detect significant changes, but for those that do, changes to arc usage are local to individual states and thus not affected by the global probability of the structure.

7.2.3 Evaluation of Mining in Non-Stationary Environments

Table 7.2 shows detection of a sequence of changes introduced to probabilities and structures in the model, beginning with the ground truth model (Figure 7.1):

1. Change of arc probabilities in XOR split structure B, from $\delta_A(q_2, b, q_3) = 0.9$ and $\delta_A(q_2, c, q_8) = 0.1$, to $\delta_A(q_2, b, q_3) = 0.1$ and $\delta_A(q_2, c, q_8) = 0.9$.
2. Change of arc probabilities in B, back to $\delta_A(q_2, b, q_3) = 0.9$, $\delta_A(q_2, c, q_8) = 0.1$.
3. Change of arc probabilities in parallel split structure C, from $\delta_A(q_3, d, q_4) = 0.8$, $\delta_A(q_3, e, q_5) = 0.2$ to $\delta_A(q_3, d, q_4) = 0.2$, $\delta_A(q_3, e, q_5) = 0.8$.
4. Change of arc probabilities in C, back to $\delta_A(q_3, d, q_4) = 0.8$, $\delta_A(q_3, e, q_5) = 0.2$.
5. Change of parallel split C to exclusive choice between d or e.
6. Removal of arc $q_7 \xrightarrow{h} q_8$.

The first part of the table shows that varying numbers of traces were estimated as necessary for mining ('Sample' column). 'Detect' reports the number of new traces generated before detecting the change. In each case, the changes were discovered, with no false positives (incorrect detection of change). To correspond to the 99% confidence in mining, p-values below 0.01 were taken as significant. The Kullback-Leibler divergence between each changed and original estimate of the underlying model is shown for comparison, but there was no clear correspondence between these values and change detection.

As a comparison, we repeated the experiment mining from logs of 500 traces. All changes were again detected, but as the second part of Table 7.2 shows, in general many more traces elapsed before detection.

Since we wait before re-estimation so that the traces used for mining will all have been drawn from the changed underlying model, we can be confident that the mined

Change (Optimal Sample)	Sample	Detect	Stdev	KL	p-val
XOR split B: b, c $0.9, 0.1 \rightarrow 0.1, 0.9$	45	9.7	4.2	0.182	0.010
XOR split B: b, c $0.1, 0.9 \rightarrow 0.9, 0.1$	271	19.7	5.5	0.026	0.007
AND split C: d, e $0.8, 0.2 \rightarrow 0.5, 0.5$	45	29.3	13.7	0.192	0.004
AND split C: d, e $0.5, 0.5 \rightarrow 0.8, 0.2$	45	39.7	19.8	0.126	0.007
AND split C changed to XOR	45	35.5	9.5	0.167	0.007
XOR split E changed to Sequence	45	36.5	19.2	0.421	0.004
Change (Large Sample)	**Sample**	**Detect**		**KL**	**p-val**
XOR split B: b, c $0.9, 0.1 \rightarrow 0.1, 0.9$	500	34.2	14.3	0.013	0.034
XOR split B: b, c $0.1, 0.9 \rightarrow 0.9, 0.1$	500	21.4	9.1	0.011	0.046
AND split C: d, e $0.8, .2 \rightarrow 0.5, 0.5$	500	122.1	30.5	0.013	0.037
AND split C: d, e $0.5, 0.5 \rightarrow 0.8, 0.2$	500	142.5	15.4	0.013	0.040
AND split C changed to XOR	500	415.3	68.4	0.015	0.043
XOR split E changed to Sequence	500	116.2	51.9	0.016	0.034

Table 7.2: Results for a Sequence of Changes. 'Sample' Traces were used for Mining, Change detected in 'Detect' Iterations (averaged over 10 Experiments), 'KL' and 'p-val' record the Kullback-Leibler Divergence and Chi^2 p-Value between new and previous Estimate of Underlying Model. 'Optimal Sample' Results used the Method described for Minimal Sample size; 'Large Sample' used excessively large Samples.

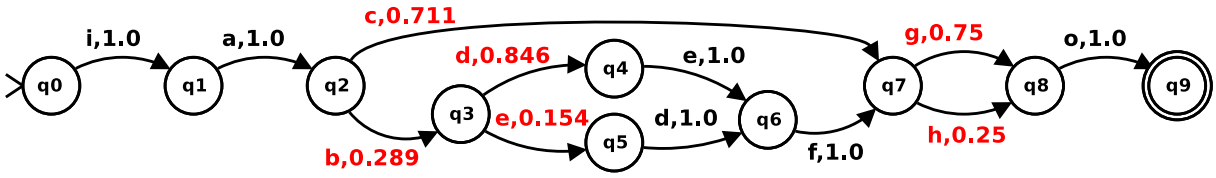


Figure 7.2: A PDFA with same Structure as Figure 7.1, but representing a significantly different Probability Distribution.

models show the sequence of changed process models over time. Figure 7.7 shows such a sequence of models, corresponding to the models in Figure 7.6 which were simulated in this experiment. We also find that in many cases change is significant but only evident in the PDFA probabilities (e.g. Figure 7.2), whereas the Petri net structure is unchanged. So for change detection, a probabilistic modelling language such as PDFA seems more appropriate than a purely structural representation.

To simulate a fast-changing environment we tried re-estimating the model without waiting after detection. This results in false detections (Figure 7.3) until enough traces have been generated for the log to reflect the new model. The initially estimated model will be invalid as the log will contain a mix of traces from the old and new models.

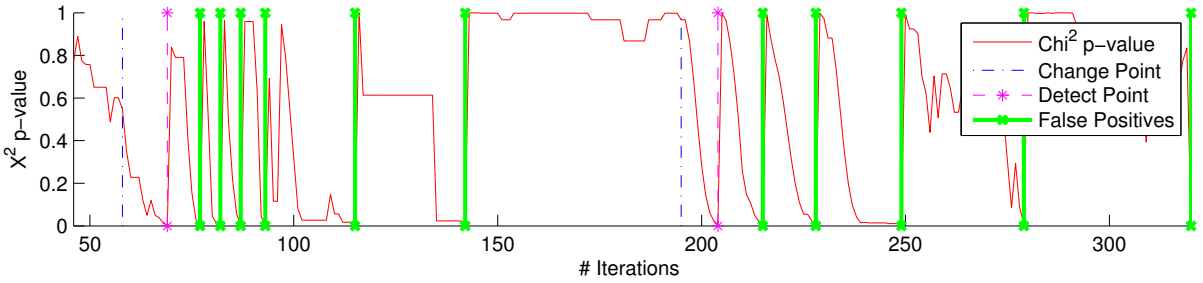


Figure 7.3: Detection of true and false Changes in Fast-changing Environment, with the Model re-estimated immediately, rather than waiting, after Change Detection.

However, although we cannot say with confidence whether these changes are valid, we can suggest that there may have been a change, and that after the n traces estimated as needed for confidence $1 - \epsilon$ in mining the correct model, have been generated, using p-value ϵ we can with confidence $1 - \epsilon$ accept the next change detected as true.

Figures 7.4 and 7.5 underline the use of the predicted minimum number of traces. With the optimal 45 traces for mining the ground truth, while no change is introduced to the generating distribution, the variation seen in the $\mathcal{X}^2$ value shows that the distributions at each iteration vary considerably. The lower graph shows that as the number of traces is increased, frequency and amplitude of these variations is reduced, with no significant (0.05) p-values. The cost is slower detection of change (Figure 7.5). Conversely, the upper graphs show that reducing the number of traces, change may be detected sooner, but less convincingly, since the $\mathcal{X}^2$ values do not converge clearly to a low value.

7.3 Chapter Summary

In this chapter we applied the probabilistic view of process mining, to online mining of processes in non-stationary environments. Since our framework allows estimation (with a given confidence level) of the number of traces needed for mining, we can mine in 'real-time' using the minimum necessary number of traces. Using statistical methods to discover change in the mined distributions, we can be confident that discovered change is true rather than an artefact of the log files, and so recover the set of changed process models

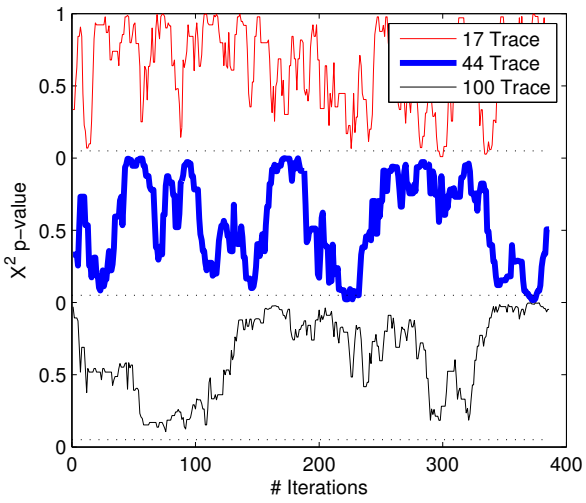


Figure 7.4: Fluctuations in X^2 p-Value over Time, from unchanged Source Process.

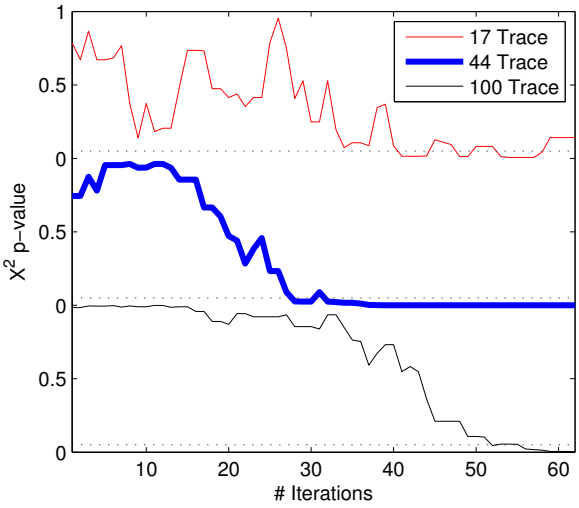
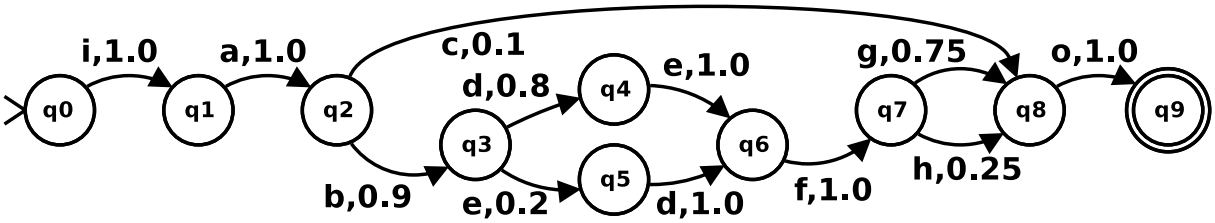


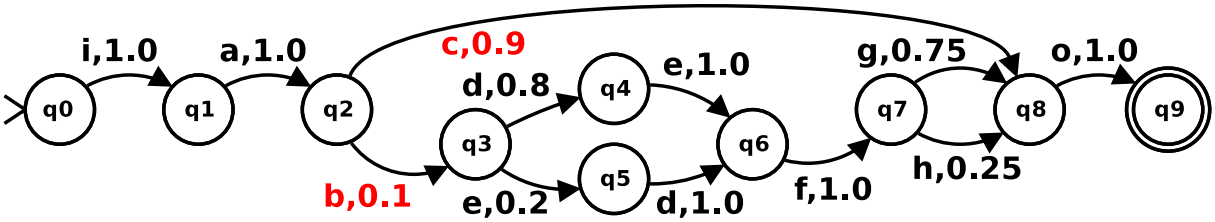
Figure 7.5: Detection of XOR Probability Change using X^2 p-Value.

in use over time. In addition, whereas process mining typically uses non-probabilistic representations such as Petri nets, this method is able to discover change that is only apparent in the probabilities in the model, while the structure is unchanged.

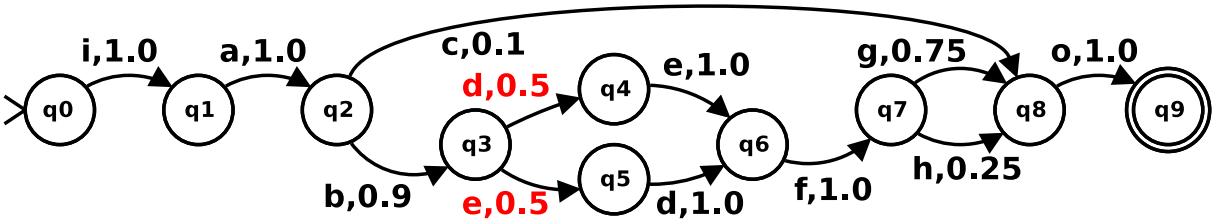
In the next chapter we consider a different practical application, that of mining from 'noisy' event logs.



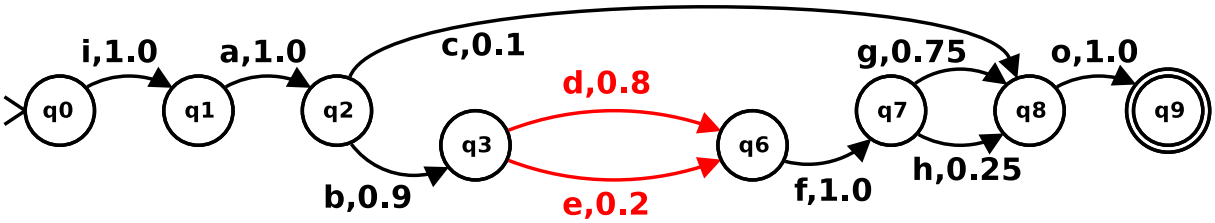
(a) Ground Truth Process



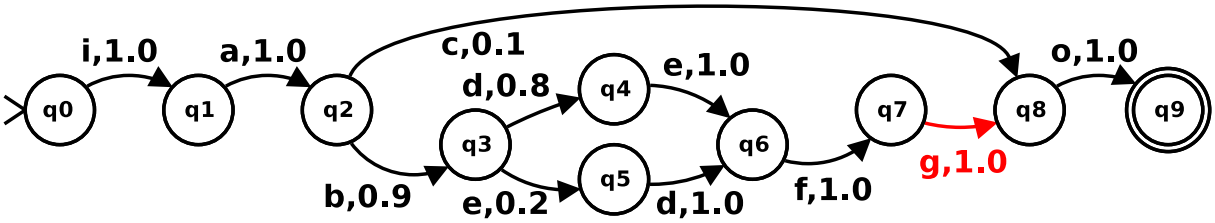
(b) Changed XOR Split Probabilities



(c) Changed Parallel Split Probabilities

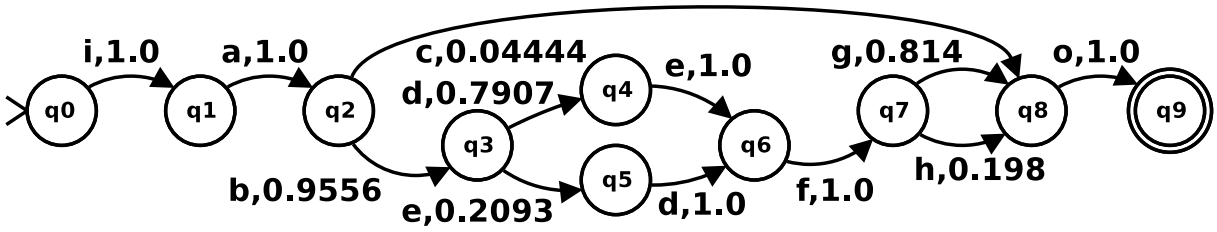


(d) Parallel Split changed to XOR

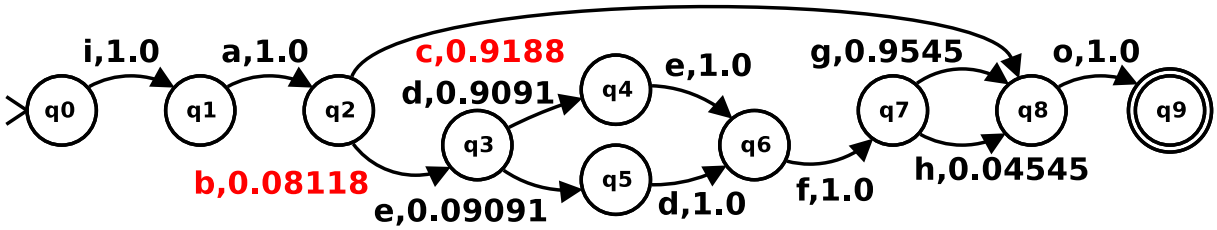


(e) XOR Split changed to Sequence

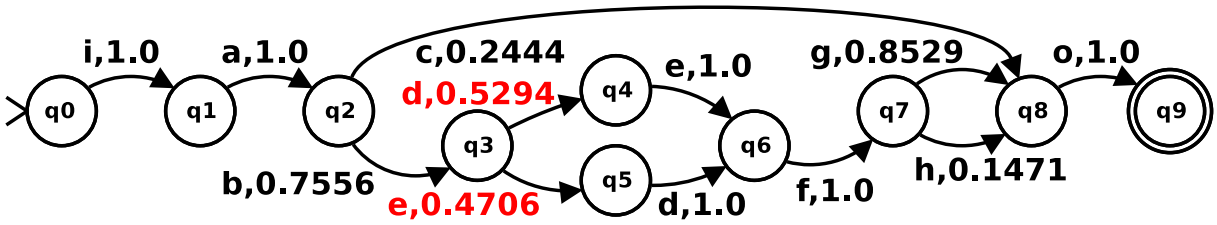
Figure 7.6: Sequence of Simulated Changed Processes corresponding to Experiments in Section 7.2.3, Table 7.2.



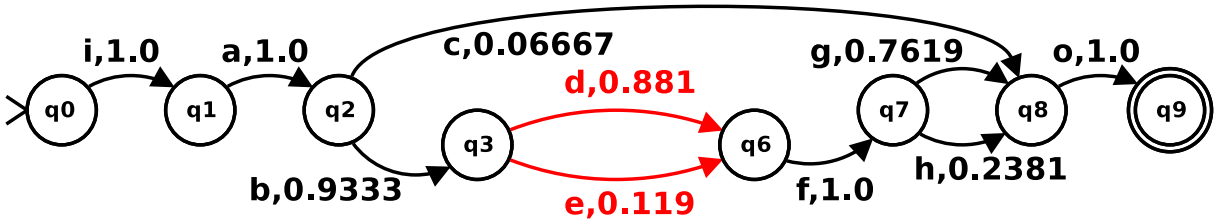
(a) Ground Truth Process



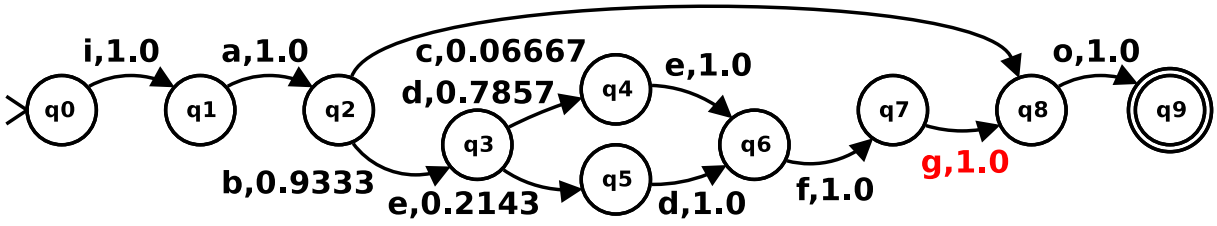
(b) Changed XOR Split Probabilities



(c) Changed Parallel Split Probabilities



(d) Parallel Split changed to XOR



(e) XOR Split changed to Sequence

Figure 7.7: Sequence of Recovered Models corresponding to Underlying Simulated Models in Figure 7.6.

CHAPTER 8

APPLICATION: PROCESS MINING FROM NOISY LOGS

In this chapter we investigate a second practical problem using our framework, that of mining from 'noisy' event logs. This is of practical importance when process mining is applied in 'the real world', since event logs often contain errors, unexpected behaviour, and may mix traces from several processes or record activity at different levels of detail.

We consider what constitutes 'noise' in event logs, and introduce one model of noise in process mining. We then extend the analysis of the Heuristics Miner [194] (Chapter 6) to understand how it is affected by noise. This leads to a method for identifying the minimum and maximum number of traces 'safe' to use for mining for a known process and amount of 'noise'.

An abridged version of the material in this chapter was first presented in [189].

8.1 Introduction

One key challenge in process mining (C6 in the Process Mining Manifesto [149]) is mining from noisy logs. In machine learning, noise generally refers to data errors such as signal error, variations in measurements, or random errors in data labels for classification. This would relate in process mining to problems in the recording of event logs. However, in process mining the term 'noise' tends to be used to refer to infrequent behaviour [148]. In

either case we face the same problem. We wish to use some of the evidence in the event log to build the process model, while ignoring other evidence, and to end up with a model of 'reasonable complexity'.

In the process mining literature there is no standard or rigorous method for defining noise in the process mining context nor its effect on the learning behaviour of algorithms. Without such a foundation, it is not possible to compare and predict the behaviour of algorithms in noisy situations. Practically, we cannot describe 'how much' noise a particular algorithm is robust to, nor how much data we should use to mine the true underlying model while excluding noise. These are important questions, since errors such as disk failure, software problems or erroneous use of systems can lead to errors in recording event logs.

In this section we introduce one formal model of noise in process mining. In Chapter 6 we described a probabilistic analysis of the Heuristics Miner algorithm [194], allowing insight to be gained into the learning behaviour of the algorithm, and prediction of the number of process traces necessary for mining to ensure that, with high confidence, a correct model will be mined. We use this analysis as a foundation to describe in Section 8.2 a model of noise in process mining and to consider the behaviour of Heuristics Miner when mining from noisy event logs.

In summary, we consider traces in event log $\mathcal{W}$ to be drawn from several underlying probability distributions, a 'ground truth' P_M (the true business process) and one or more noise distributions. For example with a noise model P_O, $\mathcal{W}$ is a sample from mixture distribution P_T,

$$P_T(t) = (1 - \kappa)P_M(t) + \kappa P_O(t),$$

such that a trace $t \in \mathcal{W}$ is drawn with a fixed probability $0 < \kappa \ll 1$ from P_O (a 'noisy' trace), otherwise from P_M. In this framework, a process mining algorithm should output a model that corresponds to P_M, rather than any convolution of P_M and P_O.

We show that upper bounds can be obtained on the number of traces 'safe' to use for mining known models, and that Heuristics Miner is more robust to some types of noise than to others. We also show that the effect of the algorithm's parameters can be predicted, allowing them to be set in an informed manner. After experimentally validating these methods (Section 8.3), we discuss in Section 8.4 the learning behaviour of Heuristics Miner, relating it to standard learning concepts. The insights gained suggest modifications that may improve the algorithm.

8.2 Effect of 'Noise' on Mining with Heuristics Miner

The most common current definition of noise in process mining is as 'outliers' or exceptional events [148, 149], i.e. parts of the process which occur infrequently. It is assumed that the mined model should not include such behaviour, which would cause a 'cluttered' or 'spaghetti' model, not useful for understanding the main process behaviour. This differs from the standard machine learning view in which noise refers to erroneous data which occurs according to some model of noise. In the context of process mining we would understand this as incorrect logging of the events that take place and their order. The justification for the current process mining view is that since an algorithm cannot distinguish incorrect logging from exceptional events, true noise (data errors) should be cleaned from the log using expert input, prior to process mining, so that the mining algorithm can assume that the log reflects what really happened.

Similarly, Fahland *et al.* [59] consider that a log may be partitioned into three sets of traces, (i) those supported by an external model, (ii) those not supported because the model is incorrect, and (iii) those not supported because they represent 'noise'. An expert, or prior knowledge, is required to identify this partition and remove the noisy traces. The cleaned log is then used to 'repair' a pre-existing model.

Historically, noise has been viewed both as errors in executing or recording the process, and as infrequent behaviour. Process mining algorithms attempt to deal with it by using

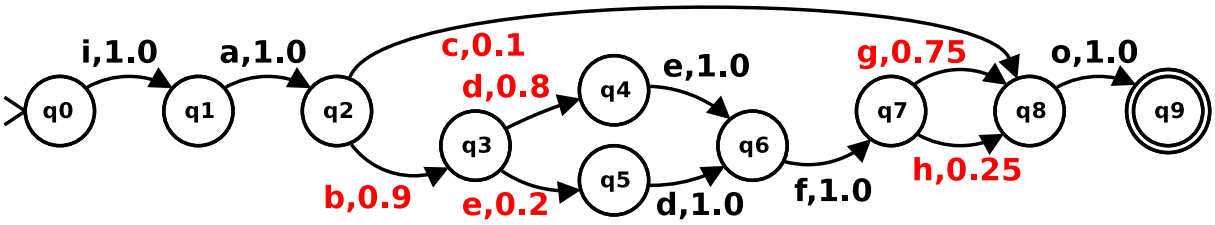


Figure 8.1: Running Example Order Process as Probabilistic Automaton, supporting Distribution P_M.

thresholds to prune input data, parts of the model or internal representations [7,38,64,70]. Agrawal *et al.* [7] note the difficulty of setting thresholds, and the sensitivity of the final model to the settings. Methods based on machine learning or optimisation techniques, such as Genetic mining [46] or Inductive Logic Programming [67] are inherently robust to noise but do not explain or predict its effect. The Heuristics Miner literature [193,194] similarly views noise as external influences on the event log, causing five types of errors: deletion of the (i) head or (ii) tail or (iii) part of the body of a trace, (iv) removal of one event, or (v) interchange of two randomly chosen events from a trace. No model of noise generation is considered.

We next describe a more formal view of noise in process mining, which covers both errors in recording events, and infrequent behaviour.

8.2.1 A Model of 'Noise' in Process Mining

As introduced in Section 8.1 we consider traces in $\mathcal{W}$ as drawn from a mixture of a 'ground truth' distribution, representing the true business process, and one or more noise distributions. To illustrate with the running example, P_M is the distribution over traces representing the true business process, represented by the probabilistic automaton in Figure 8.1. We then consider a single distribution P_O over all 'noise' traces. P_O could allow any activity to occur at any time with equal probability, or (more likely) only certain types of noise are possible.

$\mathcal{W}$ is then a sample from a mixture distribution P_T over traces, which is a convex combination of P_M and P_O, such that any trace $t \in \mathcal{W}$ is drawn with some fixed probability

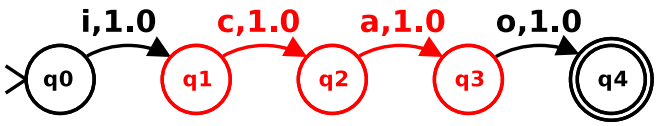


Figure 8.2: Simple Noise Model $\mathcal{O}_1$, in which Activities a and c are swapped.

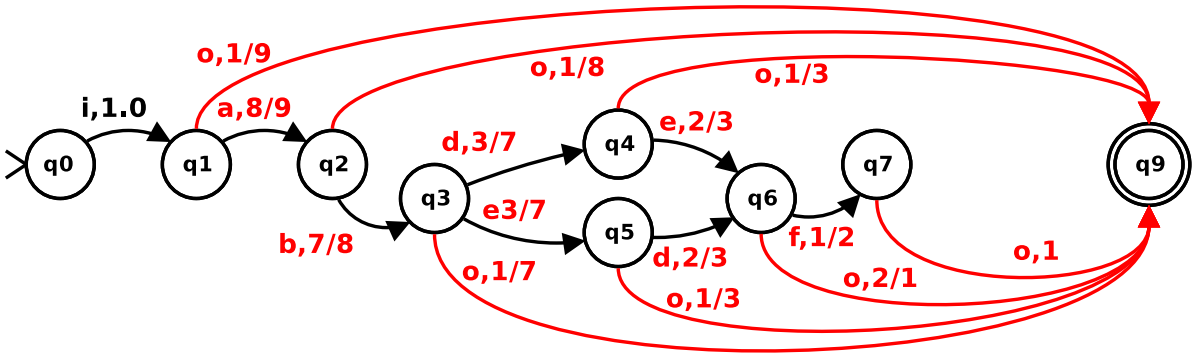


Figure 8.3: Simple Noise Model $\mathcal{O}_2$. Any Activity can be followed by o.

$0 < \kappa \ll 1$ from P_O (a 'noisy' trace), otherwise from P_M:

$$P_T(t) = (1 - \kappa)P_M(t) + \kappa P_O(t). \tag{8.1}$$

We would like our process mining algorithm to output a model that supports P_M, rather than any convolution of P_M and P_O.

This is just one possible model of noise. but having such a formal model allows investigation of a process mining algorithm to answer questions such as: *For what proportion and types of noise will the algorithm be robust? What is the effect of the noise on the amount of data needed to be confident in the accuracy of the mined process models?*

A simple example of noise caused by the process being followed (or recorded) incorrectly is the model $\mathcal{O}_1$ (Figure 8.2). The only 'noise' trace introduced by this model is *icao*, in which activities c and a have been swapped. This means that occasionally, an order is rejected before the stock is checked. A second example is model $\mathcal{O}_2$ (Figure 8.3), which allows a trace to end (with activity o) after any activity. This could illustrate systems failures (such as disk failure), which might occur with low probability at any point in the process. This model introduces traces such as $iao, iabo, iabdo, \ldots$, and so on.

We define the support of the ground truth model as a set of traces T_M, the support

of the noise model as a set T_O of 'unexpected' traces, and the traces in $\mathcal{W}$ as a set T_W:

$$T_M = \{t | P_M(t) > 0\} \subseteq \Sigma^+, \qquad T_O = \{t | P_O(t) > 0\} \subseteq \Sigma^+. \text{ and}$$
$$T_W \subseteq T_M \cup T_O, \text{ where } T_M \cap T_O = \emptyset.$$

T_W is only a subset of the union of T_M and T_O since the log is only a sample, and may include not all the traces supported by the models. T_M and T_O are disjoint because by definition any trace supported by the true model is not noise, and *vice versa*.

Since Heuristics Miner operates on pairs of activities, we define the possible pairs of activities which may occur in traces from these sets,

$$B_M = \{ab | \pi_M(ab) > 0\}, \qquad B_O = \{ab | \pi_O(ab) > 0\}, \text{and}$$
$$B_W = \{ab | \pi_W(ab) > 0\},$$

where $\pi_M(ab) = P_M(i\Sigma^* ab \Sigma^* o)$, $\pi_O(ab) = P_O(i\Sigma^* ab \Sigma^* o)$, and

$$\pi_W(ab) = (1 - \kappa)\pi_M(ab) + \kappa \pi_O(ab).$$

Again, B_W is a subset of $B_M \cup B_O$, but now $B_M \cap B_O$ may be non-empty, since the same pairs of activities may exist in traces from both the true and the noise models.

Since each trace is a sample from a mixture of P_M and P_O, noise may reduce the probabilities of pairs of activities under P_T. For all ab in B_W, $\pi_W(ab) \leq \pi_M(ab)$, because some traces in T_O may not include some pairs of activities in B_M, reducing their probability in $\mathcal{W}$. Since these probabilities influence correct mining of structures, with noise it is likely that more traces will be needed to correctly mine the structures in the process model, and thus the full model.

A model mined from $\mathcal{W}$ is also at risk of two types of problems (still assuming no cycles), described in the next subsections.

8.2.2 Introduction of Additional XOR Splits and Joins

Recall that each trace begins with the same activity i, and ends with the same activity o. Then any 'noise' trace from T_O will partially match at least one true trace from T_M, i.e. they share a common prefix x and suffix y. Let $\Sigma', \Sigma'', \Sigma'''$ partition Σ, then

$$\begin{aligned} &\forall\, t_O \in T_O,\, t_O = xvy \,\wedge\, \exists\, t_M \in T_M, t_M = xuy. \text{ where} \\ &x \in i\{\Sigma'\}^*, y \in \{\Sigma''\}^* o, u, v \in \{\Sigma'''\}^*. \end{aligned} \tag{8.2}$$

Let a be the last activity in the common prefix x, a' the first activity in u, and a'' the first activity in v. Similarly b is the first activity in the common suffix y, b' the last activity in u, and b'' the last activity in v. Then (8.2) says that for any 'noisy' trace, the first part (prefix) will match the prefix of a true trace up to a at which they diverge, and the last part (suffix) will match the suffix of a true trace after they converge at b. Between the common prefix and suffix, the noisy and partially-matching true trace differ. Since there are no repeated activities, the prefix, suffix, and non-matching part of the traces are drawn from non-overlapping subsets of the alphabet of activities.

Then we have the risk that Heuristics Miner will create an extra XOR split $a \rightarrow (a' \,\#\, a'')$ at the divergence point, or an extra XOR join $(b' \,\#\, b'') \rightarrow b$ at the convergence point. These splits and joins may be at the beginning and end of the model, when the noise trace does not match a true trace apart from the start and end activities. There may be multiple such splits and joins introduced by a trace with multiple matches.

Consider the risk of creating an unwanted XOR split. Let the true process model $\mathcal{M}$ contain a sequence $i \rightarrow a \rightarrow a'$, and the noise model introduces a pair $aa'' \in B_O$. A new arc $a \rightarrow a''$ will be created in the model mined from $\mathcal{W}$ if the Heuristics Miner requirements of Equations (3.4)–(3.6) (Chapter 6) are met, i.e.

$$\begin{aligned} &N(aa'') > PO \,\wedge\, \mathrm{DM}_{aa''} > DT \\ &\qquad \wedge \Big(|\, \mathrm{DM}_{aa''} - \mathrm{DM}_{aa'} \,| < \mathrm{RTB} \,\vee |\, \mathrm{DM}_{aa''} - \mathrm{DM}_{ea''} \,| < \mathrm{RTB} \Big), \end{aligned}$$

where e is an existing predecessor of a''. So we can use at most n' traces for mining, to keep the probability of each of these requirements below some acceptable probability $0 < \epsilon \ll 1$:

$$n' = \underset{n}{\operatorname{argmin}} \Big[\gamma_n(N(aa'') > PO) \geq \epsilon \wedge \gamma_n(\mathrm{DM}_{aa''} > DT) \geq \epsilon \\ \wedge \big(\gamma_n(|\,\mathrm{DM}_{aa''} - \mathrm{DM}_{aa'}\,| < \mathrm{RTB}) \geq \epsilon \vee \gamma_n(|\,\mathrm{DM}_{aa''} - \mathrm{DM}_{ea''}\,| < \mathrm{RTB}) \geq \epsilon\big)\Big] - 1.$$

Compare with mining from noise-free logs, where a minimum number of traces are needed for confidence greater than $1-\epsilon$ in mining a correct model. Mining from noisy logs, there is also a maximum number of traces above which confidence in mining falls below $1-\epsilon$.

8.2.3 Introduction of Parallelism

The second problem is that parallel structures may be introduced. If the pairs of activities B_O supported by the noise model include a pair (e.g. ba) that is the reverse of a pair (ab) from B_M, then HM will conclude these are in parallel if the requirements are met for mining a parallel split (Section 6.2.4). If the ground truth $\mathcal{M}$ contains sequence $i \rightarrow a \rightarrow b$, then when $\mathcal{W}$ is drawn from P_M only, DM_{ib} is zero since $\pi_M(ib) = 0$, and DM_{ab} tends to 1 as the number of traces in $\mathcal{W}$ increases, because $\pi_M(ba) = 0$. However when $\mathcal{W}$ is drawn from a mixture of P_M and P_O, and under P_O, ba has non-zero probability, DM_{ib} will tend to 1 and DM_{ab} to $d \in [0, 1]$. This introduces these risks:

1. arc $i \rightarrow b$ created because $\mathrm{DM}_{ib} > \mathrm{DM}_{ab}$,
2. arc $i \rightarrow b$ created because PO and DT requirements are met for ib, and DM_{ib} is within RTB of ab, or
3. arc $i \rightarrow b$ created because PO and DT requirements are met for ib, and DM_{ib} is within RTB of ia.

Initially these will create an XOR rather than parallel split, but nevertheless introduce a 'noise structure' into the mined model. For 'safe' mining all of these probabilities must

κ	defaults	RTB		PO		DT	
		0.01	0.1	5	1	0.5	0.95
0	84	84	84	**49**	**45** (s)	84	84
0.01	85	85	85	**49**	**45** (s)	85	85
0.05	89	89	90	**52**	**47** (s)	90	90
0.1	94	95	95	**55**	**50** (s)	94	94
0.3	122	122	122	**71**	**66** (s)	122	122
0.5	171	171	171	**100**	**95** (s)	172	172
0.01	84	84	84	**52** (s)	**52** (s)	84	84
0.05	81	81	81	**48** (s)	**48** (s)	81	81
0.1	78	78	78	**45**	**44** (s)	78	78
0.3	67	66	66	**42** (b)	**42** (b)	66	66
0.5	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)

Table 8.1: Predicted Number of Traces needed for correct mining, varying Noise κ, from $\mathcal{O}_1$ (Top), $\mathcal{O}_2$ (Bottom). Determining Factors: achieving PO Traces for Parallel Split C, except (s) achieving $\mathrm{DM}_{be} > \mathrm{DM}_{de}$, (b) mining XOR Split B.

be less than small $0 < \epsilon \ll 1$ representing an acceptable risk of the mined model being disrupted by noise, and again there is a range of traces within which mining is possible.

8.3 Experimental Evaluation with Noise

We used the methods in the previous sections to predict the effect of various amounts of different types of noise on the running example. In separate experiments, P_M was mixed with either of the noise models $\mathcal{O}_1, \mathcal{O}_2$ outlined in Section 8.2.1.

The top half of Table 8.1 shows the effect of varying κ in the mixture model (8.1), on the number of traces to with probability 0.95 successfully mine the correct model from an event log $\mathcal{W}$ sampled from a mixture of P_M and P_{O_1}. Success means mining a model which supports the traces in T_M but not those in T_{O_1}. Only the PO parameter has any effect, indicating that meeting the PO requirement for HM to decide the split is AND, rather than XOR, is the determining factor. For PO > 1, other parameters only affect the removal of the extra $d \rightarrow e$ arc, but the split will still be XOR. For PO $= 1$, $\mathrm{DM}_{be} > \mathrm{DM}_{de}$ becomes the determining factor (labelled (s)). Increasing noise reduces the probability of the valid traces, making the requirements harder to achieve.

noise	defaults	RTB : 0.01	PO : 1	DT : 0.95
0.001	6676 (r)	**18308** (r)	6676 (r)	**13058** (d)
0.002	2714 (p)	**4158** (r)	**2620** (d)	**5025** (s)
0.003	1559 (s)	1559 (s)	1559 (s)	1559 (s)
0.004	554 (s)	554 (s)	554 (s)	554 (s)
0.005	444 (s)	444 (s)	444 (s)	444 (s)
0.01	5622 (fr)	**36700** (fr)	5622 (fr)	**5878** (ed)
0.05	889 (fr)	**6372** (fr)	889 (fr)	**1178** (ed)
0.1	246 (fp)	**2466** (fr)	**237** (ed)	**590** (ed)
0.3	84 (fp)	84 (fp)	**80** (ed)	**199** (ed)
0.5	51 (fp)	**75** (fr)	**49** (ed)	**121** (ed)

Table 8.2: Predicted Numbers of Traces for affecting of the Mined Model by Noise from $\mathcal{O}_1$ (Top) or $\mathcal{O}_2$ (Bottom). Determining Factors: (s) $\mathrm{DM}_{be} > \mathrm{DM}_{de}$, (r) $|\,\mathrm{DM}_{ic} - \mathrm{DM}_{ac}\,| < RTB$, (p) $N(ic) > \mathrm{PO}$, (d) $\mathrm{DM}_{ic} > \mathrm{DT}$, (fr) $|\,\mathrm{DM}_{fo} - \mathrm{DM}_{co}\,| < RTB$, (fp) $N(fo) > \mathrm{PO}$, (ed) $\mathrm{DM}_{eo} > \mathrm{DT}$.

Repeating for model $\mathcal{O}_2$ (lower half of Table 8.1) we find counter-intuitively that increasing noise reduces the number of traces needed, until large amounts of noise are introduced. Traces from $\mathcal{O}_2$ include the pair be, required for the AND split, with higher probability than in traces from P_M. The noise traces increase $\pi_W(be)$, increasing the likelihood of correctly mining the AND split. Eventually the number of traces reduces to the point where a different structure (B) is the limiting factor (labelled (b)).

Table 8.2 shows predictions for the numbers of traces at which the different types of noise will with probability greater than 0.05 affect the mined model, above which mining is 'unsafe'. The top half of the table shows the predictions for the parallel structure $a \parallel c$ from $\mathcal{O}_1$. As expected, increasing noise reduces the number of traces safe to use. For the lowest noise, this is when DM_{ic} increases to within RTB of DM_{ac}, which causes arc $i \rightarrow c$ to be created, as PO and DT are also achieved. The risk can thus be reduced by decreasing RTB or increasing DT. With more noise, the limiting factor is $\mathrm{DM}_{ic} > \mathrm{DM}_{ac}$, which is not affected by varying parameters.

The lower half of Table 8.2 shows the maximum traces safe for mining from event logs sampled from a mixture of P_M and P_{O_2}. Since there are six possible noise structures, and mining of any one of them represents failure, we recorded the minimum number of traces

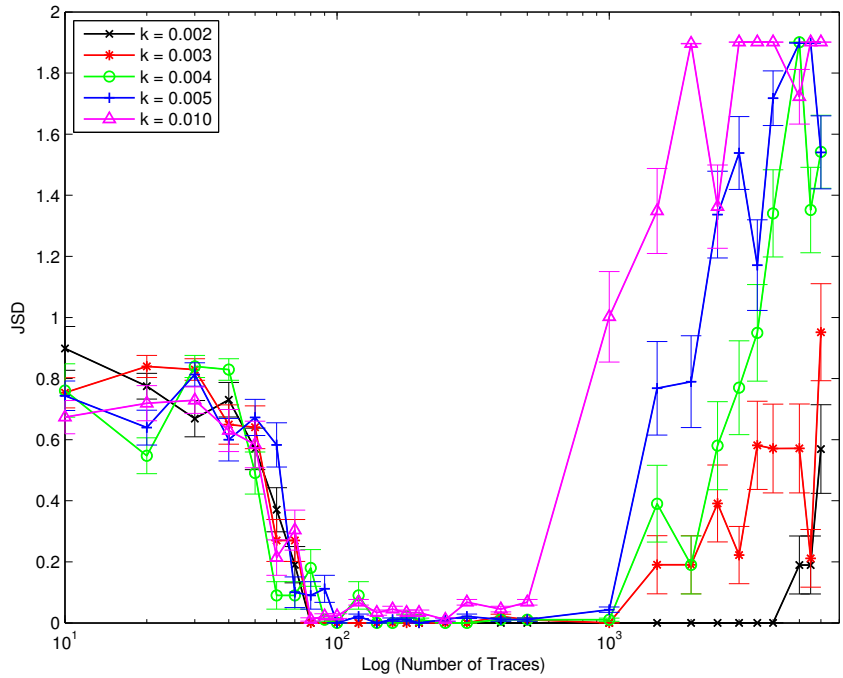


Figure 8.4: Probability of Approximately Correct Model, Mining From Logs from $\mathcal{M}$ mixed with $\mathcal{O}_1$, Various κ, Measured using JSD.

at which the probability of any one noise structure exceeded 0.05. For this model, the arcs from d, e or f to o are discovered first. For low noise, this happens when DM_{fo} is within RTB of DM_{co}, and the risk can be reduced by reducing RTB. With more noise, $N(fo) > PO$ is the limiting factor and the risk can be controlled with PO. $\mathrm{DM}_{eo} > DT$ then becomes the limit and DT can be used to control the risk of noise discovery.

These predictions were verified by simulating logs from traces randomly selected from $\mathcal{O}_1$ (similarly $\mathcal{O}_2$) or $\mathcal{M}$ according to the value of κ. Figure 8.4 shows the average distance between the ground truth and model mined from $\mathcal{M}$ mixed with $\mathcal{O}_1$ (default HM parameters, $\mathrm{DT} = 0.9, \mathrm{RTB} = 0.05, \mathrm{PO} = 10$) for various values of κ. The JSD distance is plotted against number of traces (log scale). The results confirm the predicted ranges of traces (Table 8.2). Figure 8.5 shows the results for $\mathcal{M}$ mixed with $\mathcal{O}_2$. For $\kappa \geq 0.4$, the predictions are that noise will affect the mined model before correct mining. The graph confirms that with this much noise, the JSD distance always exceeds 0.05.

Figure 8.6 illustrates effects of modifying the Heuristics Miner parameters, for noise level $\kappa = 0.1$, noise model $\mathcal{O}_2$. As predicted, reducing PO allows correct mining in fewer

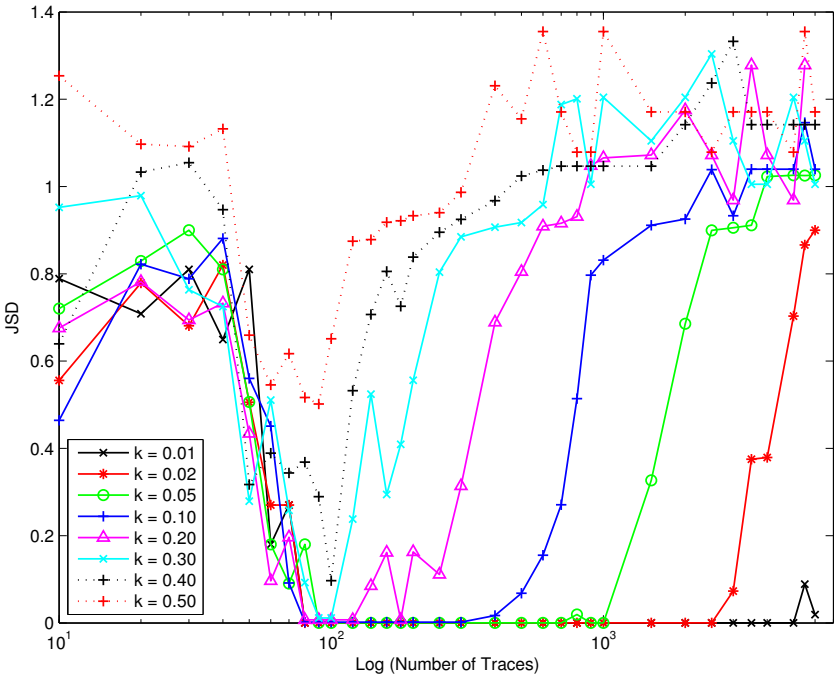


Figure 8.5: Probability of Approximately Correct Model, Mining From Logs from $\mathcal{M}$ mixed with $\mathcal{O}_2$, Various κ, Measured using JSD.

traces, since parallel splits are more easily identified as parallel rather than XOR. The effect of noise can be delayed by reducing RTB or increasing DT, making it harder for the algorithm to accept a new arc. Increasing PO has no effect in this case, but the previous discussion shows that it might be effective for other types of noise or parameter settings; there is a complex, but predictable, relationship between probabilities in the model and parameter settings.

The experimentation illustrates a general result, that Heuristics Miner is quite robust to the type of noisy traces that risk introducing additional XOR splits and joins, e.g. traces with missing heads, tails, or new traces (up to 30% for this example), but much less robust to noise that introduces parallelism, such as activities executed in the wrong order (only 1% here). The former can be reduced using the parameters (but increasing risk of omitting true infrequent arcs), but the latter is inevitable as it is discovered when the DM for a 'noise' pair of activities 'overtakes' a DM from the true model, which is not affected by parameters.

One interpretation is Heuristic Miner being most robust to systems failures, which are

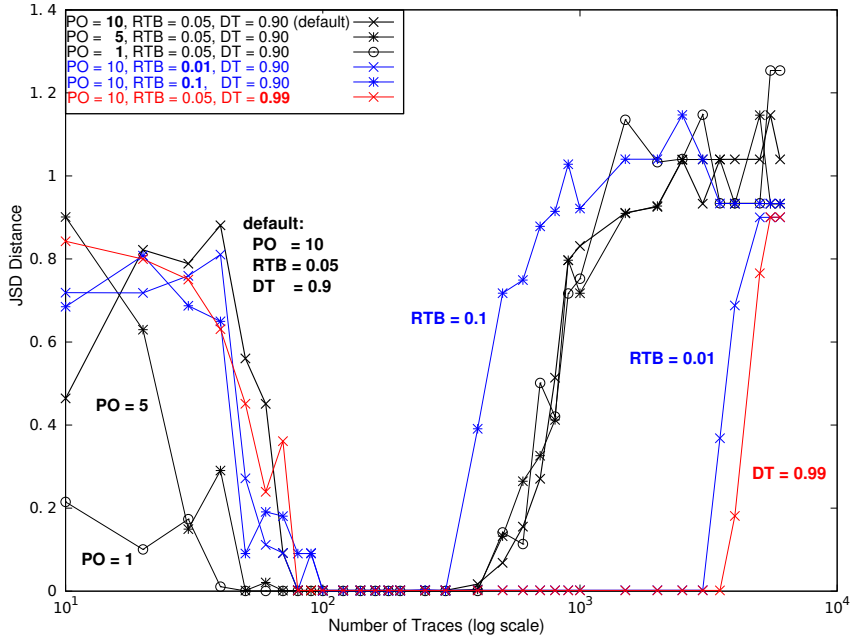


Figure 8.6: Probability of Approximately Correct Model, Various Parameter Settings, Mining From Logs from $\mathcal{M}$ mixed with $\mathcal{O}_2$, $\kappa = 0.1$.

most likely to cause partial traces, and less so to errors in the order in which participants in the business process execute or record the process.

8.4 Analysis

In this chapter we extended the results in Chapter 6 to show that for a known process, a minimum and a maximum number of traces can be identified, between which we can be confident that Heuristics Miner (HM) will mine the correct model. Below the minimum, we have too few traces to be confident in seeing enough process behaviour to mine the model (Chapter 6); above the maximum, we risk noise in the event log affecting the mined model (this chapter). It would be desirable to increase this range of traces 'safe' to use for mining, e.g. by raising the maximum. This amounts to controlling the interaction between Dependency Measures, parameters and thresholds, which controls what arcs are created (Algorithm 2). We want to optimise this interaction to ensure correct arcs are created and invalid arcs inhibited, from as wide a range as possible of size of event log.

At first glance, the behaviour of HM seems strange. When learning from a noisy

sample, using more data gives a lower quality model. In this section we look at a simple learning task which provides a loose analogy and provides insights into the learning behaviour of HM and suggests ways of improving the algorithm by increasing the maximum number of traces 'safe' to mine from, delaying the point at which noise is likely to affect the mined model.

First we describe an analogue of our notion of 'noise' in process mining. Suppose we have a sample W of data drawn from a mixture of two Gaussians, a 'Ground Truth' $P_M \sim \mathcal{N}(\mu_M, \sigma_M^2)$, and a 'noise' distribution $P_O \sim \mathcal{N}(\mu_O, \sigma_O^2)$. i.e.

$$P_T(x) = \kappa P_O(x) + (1 - \kappa) P_M(x),$$

for $0 < \kappa \ll 1$. We hope that P_M will be dominant, i.e. κ small, and we want to use W to find a single Gaussian to approximate P_M. When we take a data point from W, we want one that came from P_M but this will only be the case with probability $(1 - \kappa)$. The rest of the time we get a point drawn from P_O, representing noise. This is analogous to the process mining case, where we have two (or more) distributions generating traces, and want to find a model of the dominating distribution (the ground truth process).

This differs from common notions of noise such as additive noise. In that case, such as taking a measurement of some physical property, we assume that the measurement z_i of the true value x_i of the property, is corrupted by some random noise, e.g.

$$z_i = x_i + Y_i, \text{ where } Y_i \sim Y(\mu_Y, \sigma_Y),$$

for some distribution Y parametrised by μ_Y, σ_Y.

We want to learn P_M from n samples drawn from P_T. We use Maximum Likelihood to estimate a single Gaussian $Q(n) \sim \mathcal{N}(\mu_Q, \sigma_Q^2)$ which we hope will be close to P_M, measured by $D_{KL}\big(P_M, Q(n)\big)$, the Kullback-Leibler Divergence (KL) [89]. Maximising Maximum Likelihood is equivalent to minimising KL (e.g. [2, Defn.2.3]), so $Q(n)$ is the best estimate for P_M. As n increases, we might intuitively expect $Q(n)$ to initially approach

P_M, since samples are more likely to be drawn from P_M than from P_O. Then as n increases further we might expect samples from P_O to pull $Q(n)$ away from P_M towards P_O.

$Q(n)$ is the Gaussian that minimises $D_{KL}\big(P_T, Q(n)\big)$ [130, Theorem 3.2]. The KL Divergence between two Gaussians $P_M, Q(n)$ is defined (see e.g. [10, 111]) by

$$D_{KL}(P_M, Q(n)) = \ln \frac{\sigma_Q}{\sigma_M} + \frac{1}{2}\Big[\frac{\sigma_M^2}{\sigma_Q^2} - 1 + \frac{(\mu_M - \mu_Q)^2}{\sigma_Q^2}\Big].$$

$D_{KL}\big(P_M, Q(n)\big)$ is not dependent on n. As n increases, μ_Q, σ_Q will converge uniformly to μ_T, σ_T^2, the mean and variance of the mixture P_T, so any non-uniformity in convergence of $D_{KL}\big(P_M, Q(n)\big)$ will only be due to random effects at low numbers of samples. A sample comes from either P_M or P_O randomly, according to the Bernouilli distribution with probability parameter κ. The number of samples drawn from P_O therefore follows a Binomial distribution $N_O \sim Bin(\kappa, n)$. Above some number of samples n' the probability $M^{(n)}$ that no traces have been drawn from O will be negligible, e.g. for small $0 < \epsilon \ll 1$,

$$M^{(n)} = (1-\kappa)^{n'} \leq \epsilon \Rightarrow n' \leq \frac{\ln(\epsilon)}{ln(1-\kappa)}. \tag{8.3}$$

Thus for $0 < n < n'$ samples we expect no samples to be drawn from P_O, so the ML estimate $Q(n)$ will approach P_M. For $n > n'$, samples will be drawn from both P_O and P_M in a proportion which converges uniformly to $\frac{\kappa}{1-\kappa}$, so $Q(n)$ will approach $\mathcal{N}(\mu_T, \sigma_T^2)$, i.e. move away from P_M.

Figure 8.7 shows $D_{KL}\big(P_M, Q(n)\big)$ plotted against n for various κ, confirming the initial approach to P_M is real but only just outside standard error (see inset for $\kappa = 0.2$). For this experiment, $P_M \sim N(10, 1), P_O \sim N(5, 1)$ and results were averaged over 1000 trials. Since $M^{(n)}$ decreases exponentially (8.3), only a slight approach to P_M is possible before reaching n' and subsequently moving away. The effect is most noticeable for $k \in [0.1, 0.3]$. With more noise, for even small n the probability of drawing from P_O is too high to see any approach to P_M. With too little noise, P_O has too minor an effect on the accuracy of $Q(n)$ in estimating P_M, for any local minimum KL to show in the convergence graph.

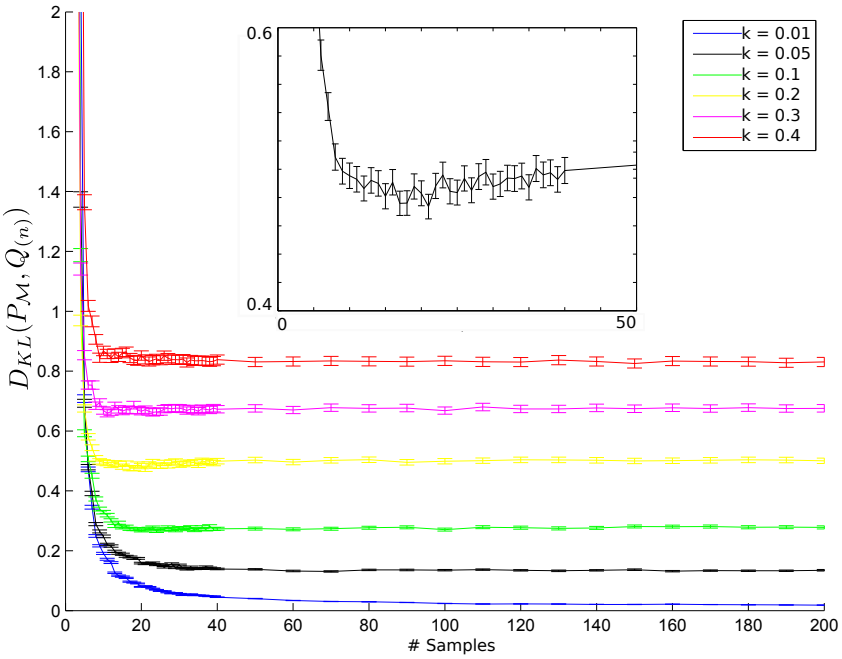


Figure 8.7: Learning Gaussians from Mixture with Various Noise κ (lowest Curve $\kappa = 0.01$, highest $\kappa = 0.4$). Inset shows Detail for $\kappa = 0.2$.

So the intuition is correct, but the effect almost negligible. We can however construct a modified algorithm (Algorithm 3) which exhibits the 'convergence/divergence' behaviour more strongly. This would be a strange way of solving this learning problem, but we will see that it has analogies with the behaviour of Heuristics Miner. Essentially, we treat the distributions as discrete, dividing the range of samples into bins, and only accept a limited number of samples from each bin.

The lower half of Figure 8.8 shows graphs for various κ. KL initially reaches a local minimum as before (Figure 8.7), but then diverges markedly. Intuitively, for each new sample either (i) previously 'unfilled' bin in P_M becomes 'full' (this is the $h+1$ sample from the bin), (ii) equivalently a bin in P_O, or (iii) no effect, sample is from a bin which is already 'full' or contains fewer than h samples. The probabilities of these events are

$$Pr(i) = (1-\kappa)U_M, \quad Pr(ii) = \kappa U_O,$$
$$Pr(iii) = 1 - P(i) - P(ii),$$

Algorithm 3 'Bins' Estimation of True Gaussian from Mixture

1: Set the range from which we expect most samples,

$$X = [a, b] \text{ s.t. } P_T(x) < \delta, \forall x < a, x > b, 0 < \delta \ll 1.$$

2: Divide X evenly into m 'bins'.
3: Draw n samples $x_i \in X$ from P_T, $1 \leq i \leq n$.
4: Count a bin as 'full' if more than h samples fall into it, for some threshold $h \geq 1$. Estimate μ_T, σ_T^2 using a representative point (e.g. the mid-point) from each 'full' bin (counting a bin only once), i.e.

$$Y = \Big\{ y_i | 1 \leq i \leq m \wedge y_i = a + \frac{i - \frac{1}{2}}{m}(b - a) \wedge |x \in X, y_i - \tfrac{1}{2m} < x \leq y_i + \tfrac{1}{2m}| > h \Big\},$$

$$\mu_Q = \frac{1}{|Y|} \sum_{i=1}^{|Y|} y_i, \quad \sigma_Q^2 = \frac{1}{|Y|} \sum_{i=1}^{|Y|} (y_i^2) - \mu_Q^2.$$

where U_M is the probability that a random sample from P_M is from a 'bin' with exactly h samples, likewise U_O. For low n, less than some n', $Pr(i) > Pr(ii)$, since $(1 - \kappa) > \kappa$, and $U_M > U_O$ because more bins cover P_M. $Q(n)$ approaches P_M. As samples are accepted from P_M, U_M reduces faster than U_O as bins 'fill', until from some $n = n'' > n'$, $Pr(i) < Pr(ii)$ and $Q(n)$ moves away from P_M towards P_T. Eventually all bins are 'full' and $Pr(i) = Pr(ii) = 0$. Divergence from P_M is apparently worse than for the first algorithm; we seem to have lost something.

But the top half of Figure 8.8 is more interesting. $D_{KL}\big(P_M, Q(n)\big)$ is plotted against n for various values of the threshold parameter h. Increasing h delays reaching minimum Kullback-Leibler Divergence, but the minimum is lower and crucially, divergence from this minimum is delayed. So increasing h sacrifices speed of learning and quality of the model learned from an 'infinite' sample, but increases the quality of the model learned from a carefully selected sample size.

Algorithm (3) is a toy example which provides insights into the learning behaviour of Heuristics Miner. HM learns discrete components (arcs and process structures) of a model which we consider to represent a distribution over traces. These components are learned discretely at various thresholds of numbers of traces as the Dependency Measures become

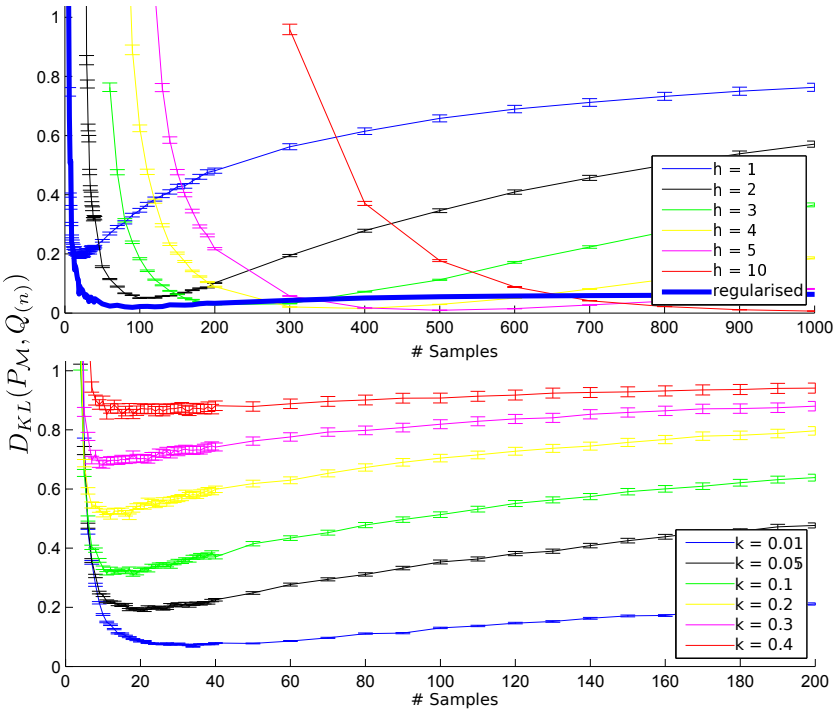


Figure 8.8: Results of 'Bins' Learning Algorithm. Bottom: Various Noise κ (Lowest Curve $\kappa = 0.01$, Highest $\kappa = 0.4$). Top: Various Thresholds h (Leftmost $h = 1$, Rightmost $h = 10$). The Lowest, Heavy-Weight Curve shows Results with a Naïve Regularisation (see Text).

ordered correctly. With more data we expect more correct structures in the mined model, but additional data does not change the structures that are already learned (approaching P_M) – until noise starts affecting them (moving away from P_M to a noisy model).

Figure 8.8 is illustrative of the behaviour of the HM parameters, tuning of which can assist learning by increasing the range of sample sizes for which successful learning can be expected (*cf* Table 8.2). However, noise always eventually affects the model. It would be desirable to 'regularise' the learning in some way so that good convergence would be achieved without subsequent divergence. The graphs are reminiscent of the 'bias-variance tradeoff' (e.g. [22]). For example when training a Neural Network, an overly complex model can be avoided by stopping training early or by using a regularisation parameter to limit changes to the model parameters. Our method for predicting the amount of data for mining correctly without being affected by noise is equivalent to early stopping since it enables use of the optimal amount of data, where too much would overfit the model.

As an illustration of regularisation, we add the following heuristic to algorithm (3). Retain μ_1, σ_1 as the estimates of the mean and standard deviation of $Q(n)$ obtained from a small sample ($0 < n \ll n'$), which should be reasonably unaffected by noisy samples (although not a good final estimate). Then accept only subsequent samples which are not too different from these estimates, e.g. $x_i \in [\mu_1 - 2\sigma_1, \mu_1 + 2\sigma_1]$. The heavy line in Figure 8.7 shows that this results in successful convergence with very little subsequent divergence. Essentially we use prior knowledge to bootstrap the parameters, to restrict the hypothesis space and the global properties of the sample learned model (see for example [86]).

This suggests that some sort of regularisation could be used to improve the robustness of Heuristics Miner when mining from noisy logs. Parallelism (model $\mathcal{O}_1$) is introduced when an unwanted $\mathrm{DM}_{ic} > \mathrm{DM}_{ac}$. This might be mitigated by limiting the maximum value of a DM to some $0 \ll d < 1$, i.e.

$$\mathrm{DM}_{ab} = \min\Big(\frac{N(ab) - N(ba)}{N(ab) + N(ba) + 1}, d\Big).$$

Extra XOR splits (noise model $\mathcal{O}_2$) would only be delayed by this. Instead, modifying PO to describe a fraction of the number of traces in the log rather than an absolute number, would regularise learning by ensuring that larger logs required more positive observations to evidence new behaviour. Clearly these changes may have other effects, and merit further research.

This discussion shows the importance of understanding the learning algorithm, predicting amount of data, and informed setting of parameters

8.5 Chapter Summary

In this chapter we explored a second practical application of our probabilistic framework for considering process mining. We presented a formal model of noise in process mining in which traces in an event log are generated at random by a mixture model consisting

of a distribution over traces from the true process, mixed with one or more distributions over erroneous traces. Algorithms should mine a model supporting only the traces from the true distribution.

We used the analysis of the Heuristics Miner algorithm [194] in Chapter 6 to investigate the ability of this algorithm to handle noise. This gave insight into the effects of different types of noise on the algorithm's behaviour, and into the effect of varying its parameters. Heuristics Miner is seen to be quite robust to the type of noise that risks additional XOR splits and joins, but less so to noise introducing parallelism. This is a general result, since the former can be controlled using the parameters of the algorithm, at the risk of omitting true low-probability arcs, but the latter will always affect the model once the 'noisy' Dependency Measure grows bigger than the true one. This is not affected by the parameter settings, unless the UH parameter is unset.

We related these findings to standard machine learning concepts, showing that this type of analysis can provide useful insights into the behaviour of process mining algorithms and provide guidance for making improvements. Practically, this work provides a method to mine 'safely' in the presence of 'noise', and to support informed setting of parameters.

This chapter brings to a close the theoretical and practical contributions of this thesis. The practical examples show that the theory presented in Chapter 4 provides a useful framework within which to successfully address real problems in process mining. In the next chapter we critically evaluate the work which has been presented, and discuss the ways in which it may usefully be extended.

Part III

Evaluation

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

The main contribution of this thesis is a probabilistic framework within which to consider process mining and analyse process mining algorithms. We presented the framework in Chapter 4, then validated it by application to analyses of two fundamental process mining algorithms (Chapters 5 and 6), and investigation of two practical and current process mining questions (Chapters 7 and 8). We showed the applicability of the framework to understanding the behaviour of process mining algorithms, predicting the amount of data needed for successful mining from both 'noise-free' and 'noisy' process event logs, and mining in both stationary and online (changing) environments. In this chapter we critically evaluate this work and suggest ways in which it may be extended.

9.1 Evaluation of the Framework

We first summarise and evaluate the main contributions of this thesis.

9.1.1 Theoretical Contributions

Many process discovery algorithms assume complete logs or only recreate the behaviour in the log, and do not recover model probabilities. However, real processes are probabilistic, so a log is only a sample of the true behaviour. The amount of data needed to be confident in mining depends on the underlying distribution, and on the behaviour of the algorithm.

We discussed process mining from a machine learning viewpoint, and introduced a probabilistic framework for considering processes and mining algorithms. We proposed that the primary task of mining the control-flow of the process is to learn the ground truth distribution over process traces, from a finite random sample of process traces drawn from the ground truth. Process mining algorithms secondarily address additional requirements such as the representation language to use, abstraction from detail, and so on. Within this framework, process models may be compared using distances between the distributions which they generate, rather than representation-dependent methods, and the behaviour of algorithms considered in terms of their convergence to the ground truth.

We analysed the well-known Alpha and Heuristics Miner process mining algorithms under this framework (Chapters 5 and 6), confirming known behaviours of the algorithms, and giving additional insights. Alpha is formally proven to correctly mine models whose underlying process is representable by a Structured Workflow Net [156], but only from noise-free logs. It uses a few simple rules to derive relations between pairs of activities observed together in an event log, and constructs a Petri net using these relations. Our analysis derived a correspondingly small set of formulae for the probability of correct discovery of these relations and process sub-structures. The behaviour of Alpha is thus predictable under our framework. Using the derived formulae we confirmed and explained the inability of the algorithm to mine correctly from noisy event logs. Our method could not be applied in its current form to unstructured processes, but such processes are not mineable by Alpha.

Heuristics Miner similarly uses relations between pairs of activities observed together in an event log, but instead of making hard decisions, uses frequencies and relative frequencies of observations of pairs of activities in the event log. These are combined with threshold parameters to allow user control of construction of the model. This gives flexible control of the detail in the mined model, but complicates the analysis under our framework. Formulae can still be derived, to bound the amount of data needed for successful mining. Heuristics Miner is differentiated from Alpha by its tolerance of noise in event

logs, and its use of parameters [194]. Our framework allowed analysis of both of these characteristics, enabling us to show and quantify that Heuristics Miner is still susceptible to noise, and to gain insights into how the tolerance to noise might be improved. The analysis, and the ability to analyse under this framework, confirm Heuristics Miner as more practical than Alpha in 'real-world' applications (e.g. [35, 45, 67, 99, 152, 191]).

Although we investigated only two algorithms, and limited our analysis to the control-flow of processes, our probabilistic approach is general, Any process mining activity that uses an event log is learning from a random sample, from underlying aspects of the true process. Any mining algorithm can in principle be analysed probabilistically in terms of how it uses the data in the log and the rules it applies to construct a process model. However, some algorithms (e.g. Heuristics Miner) can be difficult to fully analyse in this way. A simple algorithm may mask complex probabilistic behaviour. The usefulness and ease of application of our approach will be improved by developing more general approaches to understand the behaviour of types of algorithms, or use key characteristics of a process model to give bounds for the amount of data to use for mining.

9.1.2 Practical Contributions

The first practical application of this probabilistic view of processes and mining algorithms is the ability to estimate, to a given confidence level, the number of traces needed for mining. This is an important question (see Chapter 3 and [26,175,197]). We demonstrated the utility of our method using several example processes (Chapters 5 and 6). We also demonstrated in Chapter 5 how convergence graphs, from mining from event logs of increasing size, can yield insights into the learning behaviour of an algorithm. This could be applied to estimating the number of traces needed for mining the core behaviour of, or significant sub-structures in, a process, and thus to applying process mining to Automated Case Management (ACM) [140], which deals with less structured processes.

Our framework provides a foundation on which to address practical applications of current importance in process mining. We investigated mining from changing processes

(Chapter 7) and from 'noisy' event logs (Chapter 8), both mentioned in the Process Mining Manifesto [149]. We presented a novel method for online mining of processes in non-stationary environments, showing how to efficiently recover the sequence of significantly different process models over time, where differences may be in either structure or probabilities. Previous methods [26] to detect changed processes were not able in a principled manner to assess significance of change nor to detect change online.

Chapter 8 introduced a formal model of noise in process mining and compared it with current definitions of noise [59,148,193,194]. This model is general. Although we used only a single noise distribution, the event log could contain traces from a mixture of multiple models. We showed that this model of noise provides a sound basis for understanding the range of process traces 'safe' to use for mining. The analysis also showed that a formal approach can counter incorrect intuitions about an algorithm's behaviour. One might expect that using more (noisy) data would result in a better model, since for example when estimating a measurement subject to some error, averaging repeat measurements will give a more accurate result. Process mining literature refers to the 'completeness' of logs affecting the ability of algorithms to mine (see Chapter 3). Our analysis however showed that in the presence of noise, using more data is not sufficient to ensure correct mining. The process mining task is more complex than estimating a measurement, as are the algorithms involved. Completeness itself is a problem when the data is noisy.

9.2 Assumptions, Limitations and Criticisms

The main assumption underlying this work is that the control-flow of a business process can be considered as a distribution over sequences of activities. Traces in the event log are assumed to be identically and independently drawn (*i.i.d.*) from an unchanging underlying distribution. The discussion of online process mining in Chapter 7 retained this assumption, i.e. that the process is stationary between changes. This assumption enabled the analysis and prediction discussed in the preceding chapters.

It could be argued that this assumption is not always valid. For example, the probabilities of following a cycle may reduce as the number of iterations increases. Or there may be structure in the distribution: perhaps a worker is likely to follow the same sequence of actions each time they follow the process. We have taken the frequentist approach of assuming that in the long run, the process is approximately random, but it would be interesting to investigate the use of more expressive probabilistic models to model such structure and time dependency in processes. Little work has been done in this area, although History-Dependent Stochastic Petri Nets have been proposed for prediction of process outcomes [134] based on the history of traces, and some algorithms (e.g. [169,195]) model longer-distance dependencies in the data, relaxing the Markov assumption that the probability of an activity depends only on the previous activity. However, since process mining exists to support the business community, and understandable representations are preferred, such more expressive models may be too complex to be of practical use.

Our analysis also assumes that the models produced by process mining algorithms can be converted to probability distributions. This is not always the case, for example in [141] an algorithm and representation is presented which loses the distinction between exclusive and parallel splits. Such cases represent particular business requirements, and would necessitate further research to understand how to apply our framework and analyses.

As discussed in Chapter 1, our analysis is limited to acyclic processes, which simplifies the analysis since process distributions have finite support. We also assume, as elsewhere in the literature, e.g. [7, 39, 156], processes with single start end activities, atomic activities which are recorded as they occur, and algorithms which make no use of additional information such as timing. These assumptions do not limit the framework, which could be relatively easily extended to encompass a wider set of process models and algorithms.

A more serious limitation of our method is that the process of analysing an algorithm is difficult and needs to be applied from scratch to each algorithm. The analysis of Heuristics Miner in Chapter 6 showed that a relatively simple algorithm can mask complex probabilistic behaviour. More complex algorithms such as the Genetic Miner [50] may

prove impractical to analyse in this way. We believe that on the basis of this work, more general approaches can be developed to understand the behaviour of particular types of algorithms, and to use key characteristics of, or limited information about, a process model or algorithm to predict the amount of data needed for mining.

In the same way, to understand algorithms' behaviour, we assume a known process model, and investigate the amount of data needed to successfully mine that model. Process mining is a practical activity, and one of its main benefits is the ability to mine an event log from an unknown process to gain an initial insight into the underlying process. Our framework, together with the analysis of existing algorithms to develop knowledge about general types of algorithm, could be extended to give methods for assessing the confidence to be placed in the results of mining from logs from unknown processes.

A criticism that has been levelled at this work is the use of probabilistic automata, which impose only a weak representational bias on a process model, whereas business process models tend to be structured. Probabilistic automata also do not succinctly represent parallel behaviour. However, we do not advocate the use of automata for mining. In fact, our framework is representation-independent. Probabilistic automata provide a convenient working representation for the distributions which we model, as a 'lowest common denominator' to which other representations may be converted. They also lead to useful methods for efficient comparison of distributions (e.g. [41, 42]).

9.3 Future Work

The work presented in this thesis may be developed in several ways.

9.3.1 Broadening the Scope

Initial future work should seek to remove the dependency of the framework on the assumptions in the previous section. This, together with analysis of more and different types of process mining algorithm, would fill out the gaps in the framework, allowing for processes

with cycles, activities with duration, and making use of other data attributes associated with activities and processes (see for example [141, 190]). Rather than simply repeating the analysis per algorithm, work should focus on simplifying the prediction mechanism and developing general theory for understanding the behaviour of mining algorithms.

9.3.2 Deepening the Theory

We have considered process mining within a Probably Approximately Correct (PAC) framework [147]. Future work should attempt to develop bounds within this framework, that are significantly simpler and easier to compare theoretically than the exact analytical formulae which we have presented for specific algorithms. Such bounds should be generally applicable to classes of algorithm. This would enable objective comparison of any algorithms and better understanding of what factors affect algorithms' behaviours in various situations. Ultimately, this would lead to general principles and learning theory about the capabilities of different classes of process mining algorithm.

In general, real-world processes are complex, and the visual results of process mining difficult to understand [148]. While this can be addressed by abstraction and clustering methods (e.g. [73]), preprocessing, extracting multiple processes from an event log (e.g. [24,70]), work patterns may simply be flexible, as assumed by Adaptive Case Management (ACM) (e.g. [140]). Our framework would be made more generally applicable by explicit extension to analysing core process behaviour, or of the structured parts of processes.

A machine learning view of process mining, applied to existing algorithms, may give insight into how algorithms may be improved. We showed (but did not test) an example for the Heuristics Miner, in Chapter 8. The Alpha algorithm is only guaranteed to successfully mine models whose underlying process is representable by a Structured Workflow Net [156], but the algorithm does not use this knowledge to limit itself to this hypothesis class. Incorporating such prior knowledge could lead to improved learning.

9.3.3 Practical Applications

While it is useful to understand how much data is needed for known models, as discussed above, process mining is often applied where the underlying model is not known. Future work should extend the methods we have proposed, to enable methods for quantifying the confidence which can be placed in the correctness of models mined in such situations.

Our initial investigation of process mining in online environments (Chapter 7) leaves many open questions, such as whether the analysis can be applied to more refined process mining algorithms, noisy log files, or complex or unstructured processes. Future research should develop methods for mining from continuously changing underlying distributions.

Finally, our framework provides a foundation on which to address other practical applications in a rigorous way. These include generalising process models in a principled manner, to unseen data, analysing the factors influencing decision points in process models [128], and predicting the outcome of active process instances [135, 165]. While many of these areas have been addressed in the literature, our framework provides a method to be statistically confident in the conclusions which are drawn.

9.4 Conclusion

This thesis proposed a framework for the analysis of process mining algorithms. We validated this framework on representative algorithms, and showed its applicability to solving real-world process mining problems. In this final chapter, we have shown that this work is a foundation on which future research may build, to develop general learning theory for process mining, allow mining algorithms to be objectively compared and selected, and provide practical solutions to current and future business problems.

APPENDIX A

PROOFS OF PROPOSITIONS AND THEOREMS

In this appendix we present details of proofs omitted from the main text.

A.1 Analysis of the Alpha Algorithm

Proofs of the propositions and theorem in Chapter 5 follow. $\gamma_n(E)$ denotes the probability that a requirement E for mining a structure, holds true in a log of n traces. For example, $\gamma_n(A)$ for set A in Figure A.1, is 'the probability that at least one trace in a log of n traces contains sub-string ab'. $P_{\alpha,n}(a >_n b)$ is the probability that Alpha infers the relation $a >_n b$ over n traces, and similarly for the other Alpha relations.

A.1.1 Proof of Proposition 1, Chapter 5

Proposition. *The probability that Alpha infers $a >_n b$ is*

$$P_{\alpha,n}(a >_n b) = 1 - \big(1 - \pi(ab)\big)^n.$$

Proof. To infer that b can follow a, at least one of the n traces must contain sub-string ab, so the relation will be discovered unless all traces do not contain ab. A single trace contains ab with probability $\pi(ab)$, so all n independent traces *fail to* contain ab with probability $\big(1 - \pi(ab)\big)^n$. □

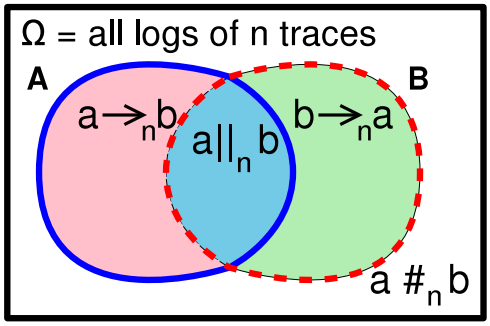


Figure A.1: The Alpha Relations on a Pair of Activities Partition the possible Logs of n traces.

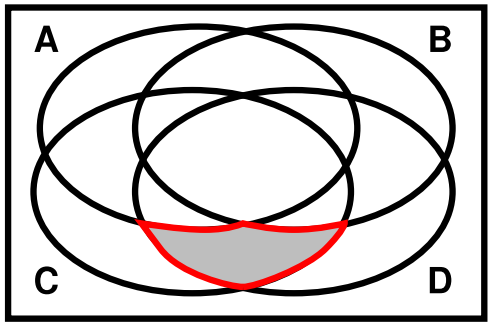


Figure A.2: Illustration of $(C \cap D) \setminus (A \cup B)$ for Proposition 5.

A.1.2 Proof of Proposition 2, Chapter 5

Proposition. *The probability that Alpha infers $a \#_n b$ is*

$$P_{\alpha,n}(a \#_n b) = \big(1 - \pi(ab) - \pi(ba)\big)^n.$$

Proof. To infer no relationship between a and b, each trace in the log must contain neither ab nor ba. This is $\neg(A \cup B)$ in Figure A.1. Since we assume no cycles, a single trace cannot contain both ab and ba, so $\pi(ab \wedge ba) = 0$. □

A.1.3 Proof of Proposition 3, Chapter 5

Proposition. *The probability that Alpha infers $a \to_n b$ is*

$$P_{\alpha,n}(a \to_n b) = \big(1 - \pi(ba)\big)^n - \big(1 - \pi(ab) - \pi(ba)\big)^n. \tag{A.1}$$

Proof. This is represented by the set $A \setminus B$ in Figure 5.1, which can be seen to be equivalent to $\neg B \setminus \neg(A \cup B)$:

$$\begin{aligned} \gamma_n(B) &= 1 - \big(1 - \pi(ba)\big)^n \quad \text{(by Prop. 1)} \\ \Rightarrow \gamma_n(\neg B) &= \big(1 - \pi(ba)\big)^n, \quad \text{and by Prop. 2,} \end{aligned} \tag{A.2}$$

$$\gamma_n\big(\neg(A \cup B)\big) = \big(1 - \pi(ab) - \pi(ba)\big)^n. \tag{A.3}$$

From equations A.2 and A.3, because $\neg(A \cup B) \subset \neg B$,

$$\begin{aligned} P_{\alpha,n}(a \ \rightarrow_n \ b) &= \gamma_n\big(\neg B \setminus \neg(A \cup B)\big) \\ &= \gamma_n(\neg B) - \gamma_n\big(\neg(A \cup B)\big) \\ &= \big(1 - \pi(ba)\big)^n - \big(1 - \pi(ab) - \pi(ba)\big)^n. \end{aligned}$$

This is intuitively interpretable as the probability of not seeing ba in any of n traces (good), minus the probability of *also* not seeing ab in any of those n traces (bad). □

A.1.4 Proof of Proposition 4, Chapter 5

Proposition. *The probability that Alpha infers* $a \ \|_n \ b$ *is*

$$P_{\alpha,n}(a \ \|_n \ b) = 1 - \big(1 - \pi(ab)\big)^n - \big(1 - \pi(ba)\big)^n + \big(1 - \pi(ab) - \pi(ba)\big)^n.$$

Proof. The relations partition the set of possible logs (Figure 5.1); thus $P_{\alpha,n}(a \ \|_n \ b) = 1 - P_{\alpha,n}(a \ \rightarrow_n \ b) - P_{\alpha,n}(b \ \rightarrow_n \ a) - P_{\alpha,n}(a \#_n b)$, following the previous results. □

A.1.5 Proof of Proposition 5, Chapter 5

Proposition. *The Probability that Alpha infers an XOR split is*

$$\begin{aligned} P_{\alpha,n}\big(a \ \rightarrow_n (b_1 \# \ \ldots \ \# \, b_m)\big) = S_n(N) - \sum_{1 \le i \le m} S_n\big(N \cup \{Y_i\}\big) \\ + \sum_{1 \le i < j \le m} S_n\big(N \cup \{Y_i, Y_j\}\big) - \ldots + (-1)^m S_n(N \cup Y), \end{aligned} \tag{A.4}$$

where

$$S_n(X) = \Bigg(1 - \sum_{1 \le i \le |X|} \pi(X_i) + \sum_{1 \le i < j \le |X|} \pi(X_i \wedge X_j) - \\ \ldots + (-1)^{|X|} \pi(X_1 \wedge X_2 \wedge \ldots \wedge X_{|X|})\Bigg)^n. \tag{A.5}$$

Proof. We begin with the probability that the pairs of tasks which *must not* be seen in the log, do indeed not occur in the log, then use the 'inclusion-exclusion principle' to remove the probability that any of the pairs of tasks which *must* be present in the log, are also missing from the log.

For events E_i in a probability space with N events,

$$\gamma_n\Big(\bigcup_{1 \le i \le N} E_i\Big) = \sum_{1 \le i \le N} \gamma_n(E_i) - \sum_{1 \le i < j \le N} \gamma_n(E_i \cap E_j) + \\ \ldots + (-1)^{N-1} \gamma_n\Big(\bigcap_{1 \le i \le N} E_i\Big). \tag{A.6}$$

As a simplified example, we consider discovery of a two-way XOR split from a to b and c, and assume $\pi(ba) = \pi(ca) = 0$. For Alpha to discover the split, the log must include ab and ac, but not bc or cb. If Figure A.2 represents the set of all logs of n traces, then let set A contain all logs which contain *no* ab, B *no* ac, C *no* bc, and D *no* cb. Then we need the probability contained in the shaded area:

$$\begin{aligned} \gamma_n\big((C \cap D) \setminus (A \cup B)\big) &= \gamma_n(C \cap D) - \gamma_n\big((C \cap D) \cap (A \cup B)\big) \\ &= \gamma_n(C \cap D) - \gamma_n\big((C \cap D \cap A) \cup (C \cap D \cap B)\big) \\ &= \gamma_n(C \cap D) - \gamma_n(C \cap D \cap A) - \gamma_n(C \cap D \cap B) \\ &\quad + \gamma_n(C \cap D \cap A \cap B) \text{ (by equation A.6).} \end{aligned} \tag{A.7}$$

$S_n(N)$ is represented by $(C \cap D)$, $S_n(N \cup Y_1)$ by $(C \cap D \cap A)$, etc. If we make no assumptions about the ground truth, then a single trace may include any of these sub-strings, so the

same approach is needed to calculate $S_n(X)$. □

A.1.6 Proof of Theorem 1, Chapter 5

Theorem. *The probability of discovery of an XOR split may be upper bounded by assuming independence between discovery of Alpha relations over n traces. The probability is over-stated but error rate decreases exponentially with increasing n:*

$$P_{\alpha,n}\big(a \rightarrow_n (b_1 \# \ldots \# b_m)\big) \leq \prod_{1\leq i\leq m} P_{\alpha,n}(a \rightarrow_n b_i) \times \prod_{1\leq i<j\leq m} P_{\alpha,n}(b_i \#_n b_j). \quad \text{(A.8)}$$

Proof. To demonstrate, we assume that an underlying model with an XOR split from a to $(b_1, b_2, \ldots, b_m)$ is followed without error, and traces are recorded without error ('noise-free'). Thus $\pi(b_1 a) = \pi(b_1 b_2) = 0$, etc. and the previous equations may be simplified. Equation A.1 reduces to $P_{\alpha,n}(a \rightarrow_n b) = 1 - \big(1 - \pi(ab)\big)^n$, and so on.

Let b_i be shorthand for $\pi(ab_i)$, and label equations (A.4) as $F(n)$ and (A.8) as $G(n)$. Equation (A.4) (discovery of multiple Alpha relations from one log *not* independent) reduces to

$$\begin{aligned} F(n) = 1 - &\sum_{1\leq i\leq m} (1-b_i)^n + \sum_{1\leq i<j\leq m} (1-b_i-b_j)^n \\ &- \sum_{1\leq i<j<k\leq m} (1-b_i-b_j-b_k)^n + \ldots + (-1)^m \big(1 - \sum_{1\leq i\leq m} b_i\big)^n, \end{aligned} \quad \text{(A.9)}$$

while equation (A.8), which assumes that discovery of the relations *can* be treated as independent, to

$$\begin{aligned} G(n) &= \prod_{1\leq i\leq m} P_{\alpha,n}(a \rightarrow_n b_i) = \prod_{1\leq i\leq m} \big(1 - (1-b_i)^n\big) \\ &= 1 - \sum_{1\leq i\leq m} (1-b_i)^n \\ &\quad + \sum_{1\leq i<j\leq m} (1-b_i)^n (1-b_j)^n - \ldots + (-1)^m \prod_{1\leq i\leq m} (1-b_i)^n. \end{aligned} \quad \text{(A.10)}$$

The error in assuming independent relations is given by $H(n) = |F(n) - G(n)|$. The first two terms of $F(n)$ and $G(n)$ cancel, leaving $(m-1)$ terms. The difference between the third terms of $F(n)$ and $G(n)$ determines the rate of decay of the error, since the absolute values of subsequent terms in $F(n)$ will be not greater than the third term. This is because the value of each term, and all b_i, will be between 0 and 1, so the terms are decreasing in absolute value. Similarly for $G(n)$ because each subsequent term is multiplied by a further factor between 0 and 1, itself decreasing exponentially. Now let

$$\begin{aligned} f_{ij}(n) &= (1 - b_i - b_j)^n, \\ g_{ij}(n) &= (1 - b_i)^n (1 - b_j)^n = \big(1 - b_i - b_j + b_i b_j)\big)^n \\ h_{ij}(n) &= g_{ij}(n) - f_{ij}(n), \\ \lambda_{ij} &= 1 - b_i - b_j, \\ \mu_{ij} &= \lambda_{ij} + b_i b_j. \end{aligned} \tag{A.11}$$

Then $h_{ij}(n) = \mu_{ij}^n - \lambda_{ij}^n$, so the error is bounded by

$$H(n) \leq (m-1)\Big[\sum_{1 \leq i < j \leq m} (\mu_{ij}^n - \lambda_{ij}^n)\Big]. \tag{A.12}$$

This is always positive, since $\mu_{ij} > \lambda_{ij}$ for all i, j; and decays exponentially in n after a maximum at relatively low n.

The rate of decay of the error is also exponential:

$$h'_{ij}(n) = \mu_{ij}^n \ln \mu_{ij} - \lambda_{ij}^n \ln \lambda_{ij}. \tag{A.13}$$

This is always negative after $h_{ij}(n)$ reaches its maximum, and decays exponentially in n, as the log factors are relatively negligible. The maximum error in $h_{ij}(n)$ is reached when

$$h'(n) = \mu_{ij}^n \ln \mu_{ij} - \lambda_{ij}^n \ln \lambda_{ij} = 0$$

and therefore the number of traces

$$n = \ln\Big(\frac{\ln \lambda_{ij}}{\ln \mu_{ij}}\Big) \Big/ \ln\Big(\frac{\mu_{ij}}{\lambda_{ij}}\Big). \tag{A.14}$$

n will be largest when $\lambda_{ij} \approx \mu_{ij}$, when the denominator of equation (A.14) tends to 0. This occurs when when the probabilities are small, as the difference between λ_{ij} and μ_{ij} is $\pi(ab_i)\pi(ab_j)$. But the discovery probability $F(n)$ or $G(n)$ will be correspondingly small, due to the second terms of $F(n), G(n)$. With the number of traces required to give only a 50% probability of discovery across all possibilities for the probabilities in a 3-way split, the difference in the number of traces predicted using $F(n)$ and $G(n)$ is negligible. □

A.2 Analysis of the Heuristics Miner Algorithm

Proofs of propositions in Chapter 6 follow.

A.2.1 Proof of Proposition 8, Chapter 6

Proposition. *For all relevant DM 'requirements',* $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ *is negatively related to* DM_{ia}*, i.e. as* DM_{ia} *increases,* $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ *decreases.*

Proof. Equation (6.15) gave the expected value of $N(iabc)$ conditional on $N(ia) = N(ia)'$:

$$\begin{aligned}\mathbb{E}_{Q_n(N(iabc)|N(ia)')}[N(iabc)] &= \mathbb{E}_{Q_n(N(iabc))}[N(iabc)] + c\big(N(ia), N(iabc)\big)\delta_{ia} \\ &= \mathbb{E}_{Q_n(N(iabc))}[N(iabc)] + c\big(N(ia), N(iabc)\big)\big(N(ia)' - \mathbb{E}_{Q_n(N(ia))}[N(ia)]\big).\end{aligned} \tag{A.15}$$

Note that in (A.15), $\mathbb{E}_{Q_n(N(iabc))}[N(iabc)] = m\pi(iabc)$, where m is the number of traces which pass through the split, and similarly for the other sub-strings. Also

$$\mathbb{E}_{Q_n(N(ab))}[N(ab)] = \mathbb{E}_{Q_n(N(iabc))}[N(iabc)] + \mathbb{E}_{Q_n(N(icab))}[N(icab)], \tag{A.16}$$

and similarly $\mathbb{E}_{Q_n(N(ba))}[N(ba)]$. Then for all valid $N(ia) \in [0, m]$,

$$\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}] \triangleq \frac{C}{D} = \frac{\mathbb{E}_{Q_n(N(ba)|N(ia))}[N(ba)] - \mathbb{E}_{Q_n(N(ab)|N(ia))}[N(ab)]}{\mathbb{E}_{Q_n(N(ba)|N(ia))}[N(ba)] + \mathbb{E}_{Q_n(N(ab)|N(ia))}[N(ab)] + 1}. \tag{A.17}$$

Following (A.15) we can write the numerator C and denominator D as follows:

$$\begin{aligned} C = {} & m\big(\pi(ibac) + \pi(icba) - \pi(iabc) - \pi(icab)\big) \\ & + c\big(N(ia), N(ibac)\big)\big(N(ia) - m\pi(ia)\big) + c\big(N(ia), N(icba)\big)\big(N(ia) - m\pi(ia)\big) \\ & - c\big(N(ia), N(iabc)\big)\big(N(ia) - m\pi(ia)\big) - c\big(N(ia), N(icab)\big)\big(N(ia) - m\pi(ia)\big), \end{aligned} \tag{A.18}$$

$$\begin{aligned} D = {} & 1 + m\big(\pi(ibac) + \pi(icba) + \pi(iabc) + \pi(icab)\big) \\ & + c\big(N(ia), N(ibac)\big)\big(N(ia) - m\pi(ia)\big) + c\big(N(ia), N(icba)\big)\big(N(ia) - m\pi(ia)\big) \\ & + c\big(N(ia), N(iabc)\big)\big(N(ia) - m\pi(ia)\big) + c\big(N(ia), N(icab)\big)\big(N(ia) - m\pi(ia)\big). \end{aligned} \tag{A.19}$$

We define some constants

$$\begin{aligned} D_1 = {} & m\big[\pi(ibac) + \pi(icba) - \pi(iabc) - \pi(icab) - \pi(ia)\big(c\big(N(ia), N(ibac)\big) + \\ & \qquad c\big(N(ia), N(icba)\big) - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big)\big)\big], \\ D_2 = {} & c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big), \\ D_3 = {} & m\big[\pi(ibac) + \pi(icba) + \pi(iabc) + \pi(icab) - \pi(ia)\big(c\big(N(ia), N(ibac)\big) + \\ & \qquad c\big(N(ia), N(icba)\big) + c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big)\big)\big] + 1, \text{ and} \\ D_4 = {} & c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) + c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big). \end{aligned}$$

And rewrite (A.17)

$$\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}] = \frac{D_1 + D_2 N(ia)}{D_3 + D_4 N(ia)}$$

Differentiating with respect to $N(ia)$,

$$\frac{\partial \mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]}{\partial N(ia)} = \frac{D_2\big(D_3 + D_4 N(ia)\big) - D_4\big(D_1 + D_2 N(ia)\big)}{\big(D_3 + D_4 N(ia)\big)^2}$$
$$= \frac{D_2 D_3 - D_1 D_4}{\big(D_3 + D_4 N(ia)\big)^2} \triangleq \frac{C'}{D'}. \tag{A.20}$$

The numerator C' of (A.20) determines the sign of the correlation of DM_{ba} with $N(ia)$. Expanding,

$$\begin{aligned}
C' = {} & m\big[c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big)\big] \\
& \times \Big[\pi(ibac) + \pi(icba) + \pi(iabc) + \pi(icab) \\
& \quad - \pi(ia)\big[c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) \\
& \qquad + c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big)\big]\Big] \\
& + c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big) \\
& - m\big[c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) + c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big)\big] \\
& \times \Big[\pi(ibac) + \pi(icba) - \pi(iabc) - \pi(icab) \\
& \quad - \pi(ia)\big[c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) \\
& \qquad - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big)\big]\Big] \\
= {} & 2m\big(\pi(iabc) + \pi(icab)\big)\Big(c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big)\Big) \\
& \quad - 2m\big(\pi(ibac) + \pi(icba)\big)\Big(c\big(N(ia), N(iabc)\big) + c\big(N(ia), N(icab)\big)\Big) \\
& \quad + \Big(c\big(N(ia), N(ibac)\big) + c\big(N(ia), N(icba)\big) \\
& \qquad - c\big(N(ia), N(iabc)\big) - c\big(N(ia), N(icab)\big)\Big) \\
\triangleq {} & C'_1 - C'_2 + C'_3.
\end{aligned}$$

Recall that $c\big(Nia), N(iabc)\big)$ is positive and all the other correlations negative. Therefore

- $C'_1 < 0$ since the first factor is positive, second negative.
- $C'_2 < 0$ only when $|c\big(N(ia), N(icab)\big)| > |c\big(N(ia), N(iabc)\big)|$ which means $\pi(icab) >$

$\pi(iabc)$. Also C' is positive only if $C'_2 < C'_1$, when $c\big(N(ia), N(iabc)\big)$ is relatively larger than the other correlations. But then $\pi(ab) > \pi(ba)$ so $\mathrm{DM}_{ba} < 0$, and $\mathrm{DM}_{ia} > \mathrm{DM}_{ba}$ is certain.

- C'_3 has relatively little effect since it does not involve m.

The derivative of $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ with respect to $N(ia)$ is therefore negative for all cases when the requirement $\mathrm{DM}_{ia} > \mathrm{DM}_{ba}$ is of interest. □

APPENDIX B

COMBINING PROBABILITIES FOR PROCESS SUB-STRUCTURES

We elaborate on the discussion in Section 5.3.3 on the effect on the formulae for discovery of Alpha relations and process sub-structures, of considering sub-structures in the model as dependent on 'previous' sub-structures. For the running example process model (Figure 4.1), the probability $P_{\alpha,n}(\mathcal{M})$ of correctly mining the full process model $\mathcal{M}$ is

$$P_{\alpha,n}(\mathcal{M}) = P_{\alpha,n}(A) \times P_{\alpha,n}(B|A) \times P_{\alpha,n}(C|B) \times P_{\alpha,n}(D|C) \times P_{\alpha,n}(E|D) \times P_{\alpha,n}(F|B,E). \tag{B.1}$$

The probabilities of sub-strings in the Alpha formulae are conditioned by the probabilities of the prefix strings leading up to those sub-strings, i.e. $\pi(ab)$ becomes $\pi(b|{\rightarrow}a)$; and only traces within which those sub-strings are expected to occur, are considered, e.g.

$$P_{\alpha,n}(a >_n b) = 1 - \Big(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\Big)^{n \cdot \pi(\rightarrow a)}, \text{ etc.}$$

This has the effect of a modest reduction to the probability $P_{\alpha,n}(S)$ of successfully mining structure S in n traces. The predicted number of traces n can also be obtained by using the 'local' probabilities within S, and dividing by the probability of traces 'reaching' the structure. We formalise this in the following proposition.

Proposition. *The number of traces n_c predicted as necessary for correctly mining process sub-structure S in a process model $\mathcal{M}$, conditional on correctly mining structures*

$A, B, \ldots$ before S in $\mathcal{M}$, is the same as the number of traces n_l predicted using 'local' sub-string probabilities in S, scaled by the probability $\pi(\rightarrow S)$ of 'reaching' S in $\mathcal{M}$.

Proof. Consider an XOR split $S = a \rightarrow (b \# c)$. Then we have $\pi(\rightarrow S) = \pi(\rightarrow a)$. The probabilities of sub-strings involved in S in the context of the model ('model' probabilities) are $\pi(ab), \pi(ac)$, etc., and the 'local' sub-string probabilities are $\pi(ab| \rightarrow a), \pi(ac| \rightarrow a)$, etc. Let the desired probability of correct mining be $1 - \epsilon$ for small $0 \leq \epsilon \ll 1$.

We consider n_M, n_l, n_c, the numbers of traces predicted as necessary for $100(1-\epsilon)\%$ confidence in correctly mining the Alpha relation $a \rightarrow_n b$, respectively using 1) 'model' sub-string probabilities, 2) 'local' sub-string probabilities, and 3) 'model' sub-string probabilities taking into account the number of traces expected to pass through S:

1. Ignoring the context of structure S in the model, using the 'model' sub-string probabilities,

$$1 - \big(1 - \pi(ab)\big)^{n_M} \geq (1 - \epsilon) \qquad \Rightarrow n_M \ln\big(1 - \pi(ab)\big) \geq \ln \epsilon$$
$$\Rightarrow n_M \geq \frac{\ln \epsilon}{\ln\big(1 - \pi(ab)\big)}. \quad \text{(B.2)}$$

   This treats mining of each structure as independent of mining of all other structures.

2. Calculating $P_{\alpha,n}(S)$ using conditional 'local' sub-string probabilities in S,

$$1 - \Big(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\Big)^{n_l} \geq (1 - \epsilon)$$
$$\Rightarrow n_l \ln\Big(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\Big) \geq \ln \epsilon$$
$$\Rightarrow n_l \geq \frac{\ln \epsilon}{\ln\big(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\big)} \leq n_M. \quad \text{(B.3)}$$

   This treats S as a standalone structure. $n_l \leq n_M$ given by (B.2) since the denominator of n_l will never be larger than that of n_M.

3. Treating S in context in the model, using the conditional sub-string probabilities,

and considering only traces which are expected to involve S,

$$
\begin{aligned}
1 - \left(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\right)^{n_c.\pi(\rightarrow a)} &\geq (1 - \epsilon) \\
\Rightarrow n_c \pi(\rightarrow a) \ln \left(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\right) &\geq \ln \epsilon \\
\Rightarrow n_c &\geq \frac{\ln \epsilon}{\pi(\rightarrow a) \ln \left(1 - \frac{\pi(ab)}{\pi(\rightarrow a)}\right)}. \qquad \text{(B.4)}
\end{aligned}
$$

Therefore $n_c = \dfrac{n_l}{\pi(\rightarrow a)}$. The same argument applies to the other Alpha relations and the formulae for mining of sub-structures. □

This is intuitive. If n_l traces would be needed to discover S if it were standalone ('local' probabilities), and the probability of a trace involving S is $\pi(\rightarrow S)$, then considering the context of S in $\mathcal{M}$ we expect on average $n\pi(\rightarrow S)$ trace of an event log of n traces to involve S. Then intuitively $n_c = \frac{n_l}{\pi(\rightarrow S)}$ traces will be needed to discover S in the context of the whole model. For example, if we have enough traces for confidence in mining a split, then having the necessary traces for the corresponding join is almost certain, since all traces that pass through the split will also pass through the join.

Using 'local' sub-string probabilities reduces the number of traces predicted for correct mining, but considering only the subset of traces in $\mathcal{W}$ that pass through S increases the predicted number of traces needed. Overall, the modified formulae result in a modest reduction in the number of traces predicted.

APPENDIX C

GAUSSIAN APPROXIMATIONS TO DISTRIBUTIONS FOLLOWED BY HEURISTICS MINER DEPENDENCY MEASURES

In this appendix we present further details of the method introduced in Section 6.2.5 to calculate $\pi(\mathrm{DM}_{ia} > \mathrm{DM}_{ba})$, assuming the distributions followed by the Dependency Measures to be approximated by Gaussians.

For simplicity consider the space of (x, y) coordinates, where DM_{ia} is plotted on the x axis, DM_{ba} on the y axis (Figure C.1). DM_{ia} is approximated by a Gaussian $g_x \sim \mathcal{N}(\mu_X, \sigma_x^2)$, DM_{ba} by $g_y \sim \mathcal{N}(\mu_Y, \sigma_y^2)$. We assume the Dependency Measures to be mutually independent, and write $g(x, y)$ for the joint density. Then transform to coordinate basis (x', y'), such that

$$x' = \frac{x - \mu_X}{\sigma_x}, \qquad y' = \frac{y - \mu_Y}{\sigma_y}. \tag{C.1}$$

The $x = y$ line is transformed to $x' = y'$,

$$x'\sigma_x + \mu_X = y'\sigma_y + \mu_Y \qquad \Rightarrow y' = \frac{\sigma_x}{\sigma_y}x' + \frac{\mu_X - \mu_Y}{\sigma_y}. \tag{C.2}$$

The point (x_1, y_1) on this line, closest to the origin is where it intersects a line x'' (Figure

C.1) orthogonal to it which passes through the origin,

$$y = -\frac{\sigma_y}{\sigma_x}, \tag{C.3}$$

The intersection (x_1, y_1) is obtained by setting the right-hand side of Equation (C.2) equal to the right-hand side of (C.3),

$$\frac{\sigma_x}{\sigma_y}x_1 + \frac{\mu_X - \mu_Y}{\sigma_y} = -\frac{\sigma_y}{\sigma_x}x_1 \Rightarrow \Big(\frac{\sigma_x}{\sigma_y} + \frac{\sigma_y}{\sigma_x}\Big)x_1 = \frac{\mu_Y - \mu_X}{\sigma_y} \Rightarrow \Big(\frac{\sigma_x^2 + \sigma_y^2}{\sigma_x}\Big)x_1 = \mu_Y - \mu_X,$$

and therefore

$$\Rightarrow x_1 = \frac{\sigma_x(\mu_Y - \mu_X)}{\sigma_x^2 + \sigma_y^2} \text{ and } y_1 = \frac{\sigma_y(\mu_X - \mu_Y)}{\sigma_x^2 + \sigma_y^2}.$$

The distance from the origin to the $x' = y'$ line (C.2), marked d in Figure C.1, is

$$\begin{aligned} d = \sqrt{x_1^2 + y_1^2} &= \sqrt{\Big[\frac{\sigma_x^2(\mu_Y - \mu_X)^2 + \sigma_y^2(\mu_X - \mu_Y)^2}{(\sigma_x^2 + \sigma_y^2)^2}\Big]} \\ &= \sqrt{\Big[\frac{(\sigma_x^2 + \sigma_y^2)(\mu_Y - \mu_X)^2}{(\sigma_x^2 + \sigma_y^2)^2}\Big]} = \frac{|\mu_Y - \mu_X|}{\sqrt{\sigma_x^2 + \sigma_y^2}}. \end{aligned}$$

To the transformed $x = y + \text{RTB}$ line (for separation by more than RTB),

$$d = \frac{|\mu_Y - \mu_X + \text{RTB}\,|}{\sqrt{\sigma_x^2 + \sigma_y^2}}.$$

Since these are affine transformations, co-linearity and ratios of vectors along a line are preserved, and the transformed joint distribution $g'(x', y')$ is a standard two-dimensional Gaussian with zero mean and unit standard deviation. Since the transformed $x = y + \text{RTB}$ line cuts this distribution, we can integrate a one dimensional Standard Gaussian to find $\gamma_n(x > y \pm \text{RTB})$.

$$\gamma_n(x > y + \text{RTB}) = 1 - \int_{x'=-\infty}^{x'=d} N(0,1)dx'. \tag{C.4}$$

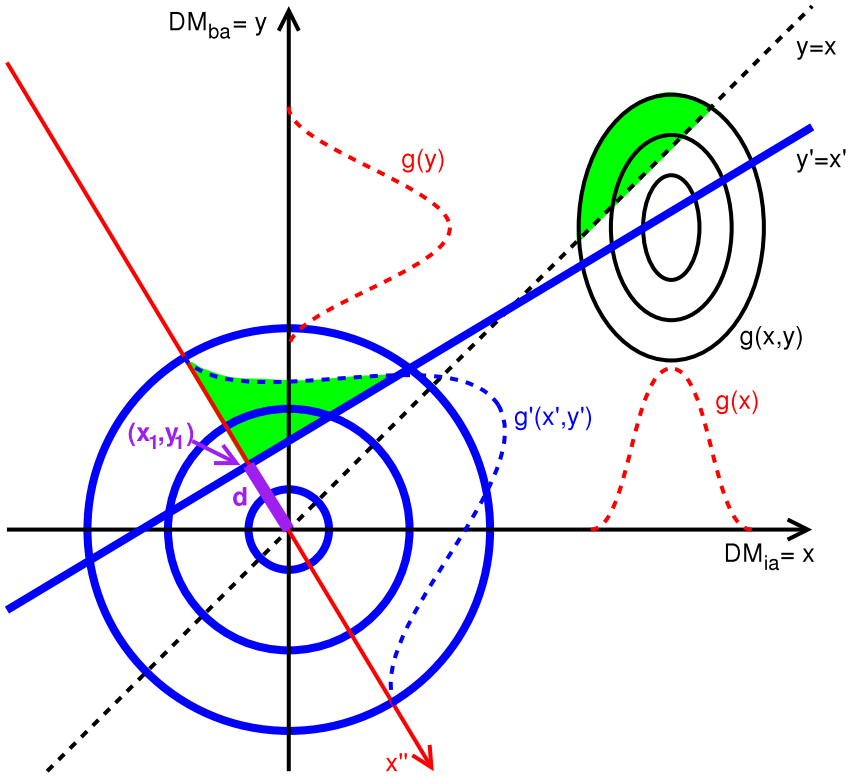


Figure C.1: Illustration of Translation and Scale of the Joint DM Distribution, $g(x, y)$, to $g'(x', y')$, to allow $\pi(\mathrm{DM}_{ia} > \mathrm{DM}_{ba} + \mathrm{RTB})$ to be calculated using the Distance d from the Origin to the transformed $x = y$ Line, $x' = y'$. See Text and Equations (C.1)–(C.4).

APPENDIX D

TABLES OF RESULTS SUPPORTING HEURISTICS MINER ANALYSIS

In this appendix we provide fuller versions of the results presented in Chapter 6.

D.1 Analysis of 2-Way Parallel Splits

Table D.1 shows the number of traces predicted for $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) \geq 0.95$ for two-way parallel splits, for all combinations of $\pi(ia) + \pi(ib) \leq 1$, in intervals of 0.05. The highlighted entries show where this is greater than $\gamma_n(N(ia) > \mathrm{PO} \wedge N(ib) > \mathrm{PO})$ (Table D.3), which can be seen to only be the case for very imbalanced splits.

Table D.2 shows the corresponding numbers of traces identified by simulation.

D.2 Analysis of 3-Way Parallel Splits

Table D.4 shows the *difference* between the predicted and simulated numbers of traces for meeting the three requirements such as $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) \geq 0.95$ for three-way parallel splits, under the assumption of 'second' probabilities proportional to the 'first', i.e. $\pi(b|ia) = \frac{\pi(ib)}{\pi(ib)+\pi(ic)}$. See Section 6.2.5 (page 123) for descriptions of these assumptions. Prediction is using the 'AND2 Method', projecting the joint distribution of $\mathrm{DM}_{ia}, \mathrm{DM}_{ba}$ to the line given by plotting DM_{ia} against $\mathbb{E}_{Q_n(N(ba),N(ab)|\,\mathrm{DM}_{ia})}[\mathrm{DM}_{ba}]$ for all values of

$\pi(ib)$	$\pi(ia) = 0.05, 0.1, \ldots, 1.0$																		
0.05	109	46	28	20	15	12	9	8	6	5	5	4	3	3	2	2	1	1	1
0.1	133	53	31	21	16	12	10	8	7	6	5	4	3	3	2	2	2	1	
	154	59	34	23	17	13	10	8	7	6	5	4	3	3	2	2	2		
	175	64	36	24	18	14	11	9	7	6	5	4	4	3	2	2			
	194	70	39	26	19	14	11	9	8	6	5	4	4	3	3				
	213	75	41	27	20	15	12	9	8	6	5	4	4	3					
	231	80	44	29	20	16	12	10	8	7	6	5	4						
	249	85	46	30	21	16	13	10	8	7	6	5							
	266	90	48	31	22	17	13	10	9	7	6								
	283	94	51	33	23	17	13	11	9	7									
	299	99	53	34	24	18	14	11	9										
	316	104	55	35	25	18	14	11											
	332	108	57	36	25	19	15												
	348	113	59	38	26	20													
	364	117	61	39	27														
	379	121	63	40															
	395	126	65																
	410	130																	
0.95	**425**																		

Table D.1: Predicted Number of Traces for $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) \geq 0.95$ for AND2.

$N(ia) \in [0, m]$, where m is the number of traces which pass through the split. Table D.5 shows the same data where 'second' probabilities are even splits, i.e. $\pi(b|ia) = \pi(c|ia) = 0.5$, etc. Negative entries indicate underestimates, that more traces were required by the simulation than were predicted, highlighted in bold. These occurs only when the split is imbalanced with one split probability being relatively small. Table D.6 shows the same data for 'extreme' imbalanced second probabilities, i.e. $\pi(b|ia) = 0.05, \pi(c|ia) = 0.95$ etc. The method underestimates significantly where one of the probabilities is very small, as discussed in Section 6.2.5.

D.3 Heuristics Miner Experimentation

In this section we provide fuller versions of the results for mining from noisy event logs with the Heuristics Miner algorithm, presented in Chapter 6.

Table D.7 presents the predicted minimum numbers of trace needed for $P_{HM,n}(\mathcal{M}) \geq$

$\pi(ib)$	$\pi(ia) = 0.05, 0.1, \ldots, 1.0$																	
0.05	94	29	19	14	11	8	7	6	5	4	4	4	2	2	2	1	1	1	1
0.1	124	46	30	14	14	8	7	6	5	4	4	4	2	2	2	1	1	1	
	124	46	30	22	17	8	7	6	5	4	4	4	2	2	2	1	1		
	153	62	30	22	17	8	7	6	5	4	4	4	2	2	2	1			
	181	61	30	22	17	8	7	10	5	4	4	4	2	2	2				
	208	61	40	22	17	14	12	10	6	4	4	4	2	2					
	208	76	40	22	17	14	12	10	7	4	4	4	2						
	234	76	40	30	17	14	12	10	7	4	4	4							
	260	89	50	30	20	14	12	10	6	4	4								
	260	90	50	30	21	14	12	10	8	8									
	286	90	50	30	22	14	12	10	8										
	291	103	50	30	23	19	12	10											
	312	103	59	37	24	19	12												
	337	103	59	37	23	19													
	360	116	59	37	24														
	381	116	59	37															
	386	118	59																
	399	128																	
0.95	**410**																		

Table D.2: Simulated Traces for $\gamma_n(\mathrm{DM}_{ia} > \mathrm{DM}_{ba}) \geq 0.95$ for AND2.

0.95 of correct mining of the running example model $\mathcal{M}$, with various amounts of noise κ and various parameter settings. Tables D.8 and D.9 give the predicted maximum numbers of traces safe to use for mining before probability is greater than 0.05 that the mined models will be affected by noise structures.

$\pi(ib)$	$\pi(ia) = 0.05, 0.1, \ldots, 1.0$													
0.05	311	311	311	311	311	311	311	311	311	311	311	311	311	...
0.1	311	154	154	154	154	154	154	154	154	154	154	154	154	...
	311	154	102	102	102	102	102	102	102	102	102	102	102	...
	311	154	102	76	76	76	76	76	76	76	76	76	76	...
	311	154	102	76	60	60	60	60	60	60	60	60	60	...
	311	154	102	76	60	49	49	49	49	49	49	49	49	...
	311	154	102	76	60	49	42	42	42	42	42	42	42	
	311	154	102	76	60	49	42	36	36	36	36	36		
	311	154	102	76	60	49	42	36	32	32	32			
	311	154	102	76	60	49	42	36	32	28				
	311	154	102	76	60	49	42	36	32					
	311	154	102	76	60	49	42	36						
	311	154	102	76	60	49	42							
	311	154	102	76	60	49								
	311	154	102	76	60									
	311	154	102	76										
	311	154	102											
	311	154												
0.95	311													

Table D.3: Predicted Number of Traces for $\gamma_n\big(N(ia) > \text{PO} \wedge N(ib) > \text{PO}\big) \geq 0.95$, for AND2, PO = 10.

$\pi(ia)$	$\pi(ib) = 0.05, 0.1, \ldots, 1.0$																	
0.05	9	6	14	6	7	9	9	3	3	3	3	9	9	7	6	2	6	9
0.1	6	11	15	8	5	15	16	3	4	3	16	15	8	4	13	12	6	
	2	15	5	16	7	7	10	10	10	10	7	5	16	5	15	2		
	6	8	16	6	13	19	4	0	4	19	13	6	16	8	6			
	7	5	5	13	15	10	14	14	5	15	13	7	5	7				
	9	13	7	19	10	15	12	15	10	19	7	13	9					
	8	15	10	4	14	12	12	14	4	10	15	8						
	3	4	12	0	10	15	14	0	12	6	3							
	3	4	12	4	5	10	4	12	4	3								
	3	6	10	19	15	19	10	6	3									
	3	14	7	13	13	7	14	3										
	9	13	7	6	7	13	9											
	9	5	16	16	4	9												
	7	8	5	1	7													
	6	15	15	6														
	3	12	3															
	6	6																
0.95	9																	

Table D.4: Difference between Predicted and Simulated Traces for Mining AND3, as (Predicted−Simulated)/Predicted%. $\pi(b|ia)$ proportional to $\pi(ib)$, etc.

$\pi(ia)$	$\pi(ib) = 0.05, 0.1, \ldots, 1.0$																	
0.05	**−5**	7	15	15	27	34	28	35	36	38	35	29	34	27	15	14	6	**−4**
0.1	7	**−2**	3	14	25	21	28	30	31	28	28	21	25	16	2	0	7	
	14	5	2	14	14	21	29	29	29	29	18	11	14	2	3	15		
	15	14	12	21	4	15	20	20	16	15	0	18	14	14	15			
	25	25	11	4	5	19	19	19	19	5	4	11	27	26				
	34	19	21	15	19	25	12	25	19	15	21	21	34					
	29	28	26	16	19	12	12	19	20	26	26	29						
	35	28	29	20	19	25	19	20	29	28	34							
	38	31	29	16	19	19	20	29	31	38								
	38	28	26	15	5	15	29	30	38									
	34	28	18	4	4	18	28	33										
	30	21	14	18	14	19	30											
	34	27	12	14	24	34												
	25	14	2	14	25													
	14	3	3	14														
	17	0	15															
	8	6																
0.95	**−4**																	

Table D.5: Difference between Predicted and Simulated Traces for Mining AND3, as (Predicted−Simulated)/Predicted%. $\pi(b|ia) = 0.5$, etc., Likely Underestimates in Bold.

0.05	6	**−8**	**−11**	**−11**	**−12**	**−13**	**−14**	**−19**	**−18**	**−21**	**−25**	**−28**	**−30**	**−36**	**−44**	**−68**	**−83**	**−107**
0.1	0	**−20**	**−33**	**−33**	**−35**	**−38**	**−40**	**−43**	**−58**	**−65**	**−69**	**−69**	**−76**	**−76**	**−105**	**−139**	**−61**	
	5	**−12**	**−49**	**−58**	**−61**	**−60**	**−65**	**−69**	**−86**	**−83**	**−88**	**−101**	**−135**	**−171**	**−131**	**−10**		
	3	0	**−46**	**−73**	**−79**	**−86**	**−86**	**−116**	**−124**	**−129**	**−147**	**−187**	**−163**	**−38**	20			
	7	8	**−30**	**−89**	**−107**	**−126**	**−138**	**−197**	**−191**	**−181**	**−161**	**−92**	**−15**	17				
	11	12	**−38**	**−77**	**−160**	**−207**	**−223**	**−196**	**−208**	**−148**	**−41**	11	30					
	7	3	**−30**	**−122**	**−154**	**−180**	**−195**	**−167**	**−100**	**−7**	24	35						
	4	5	**−24**	**−71**	**−96**	**−104**	**−86**	**−36**	**−30**	37	37							
	6	11	**−10**	**−36**	**−43**	**−29**	**−9**	17	37	35								
	8	14	**−2**	**−8**	0	8	20	36	37									
	3	15	10	**−3**	10	19	35	37										
	8	16	10	0	12	26	33											
	1	7	14	7	9	29												
	3	7	13	16	16													
	7	8	**−2**	19														
	8	5	19															
	9	2																
0.95	**−8**																	

Table D.6: Difference between Predicted and Simulated Traces for Mining AND3, as (Predicted−Simulated)/Predicted%, with $\pi(b|ia) = 0.95$ or 0.05, etc., Likely Underestimates in Bold.

κ	defaults	RTB			PO		DT			
		0	0.01	0.1	5	1	0.5	0.8	0.95	0.99
0	84	84	84	84	**49**	**45** (s)	84	84	84	84
0.01	85	85	85	85	**49**	**45** (s)	85	85	85	85
0.02	86	86	86	85	**50**	**45** (s)	87	87	87	87
0.03	87	87	87	87	**50**	**46** (s)	87	87	87	87
0.05	89	89	89	90	**52**	**47** (s)	90	90	90	90
0.1	94	95	95	95	**55**	**50** (s)	94	94	94	94
0.2	106	106	106	107	**62**	**57** (s)	106	106	106	106
0.3	122	122	122	122	**71**	**66** (s)	122	142	122	142
0.4	143	142	142	143	**80**	**78** (s)	143	143	143	143
0.5	171	171	171	171	**100**	**95** (s)	172	172	172	172
0.01	84	84	84	84	**52** (s)	**52** (s)	84	84	84	84
0.02	84	84	84	84	**51** (s)	**51** (s)	83	83	83	83
0.05	81	81	81	81	**48** (s)	**48** (s)	81	81	81	81
0.1	78	78	78	78	**45**	**44** (s)	78	78	78	78
0.2	72	72	72	72	**42**	**38** (s)	72	72	72	72
0.3	67	66	66	66	**42** (b)	**42** (b)	66	66	66	66
0.4	62	62	62	62	**49** (b)	**49** (b)	63	63	63	63
0.5	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)	59 (b)

Table D.7: Predicted Number of Traces for correct mining, varying Noise κ, from $\mathcal{O}_1$ (Top), $\mathcal{O}_2$ (Bottom). Determining Factor: achieving PO Traces for Parallel Split C, except (s) achieving $\mathrm{DM}_{be} > \mathrm{DM}_{de}$, (b) mining XOR Split B.

κ	minimum distance	defaults	RTB 0	0.01	0.1	PO 5	1
0.001	0.05	6676 (r)	**28289** (s)	**18308** (r)	**5427** (p)	6676 (r)	6676 (r)
0.002	0.05	2714 (p)	**5025** (s)	**4158** (r)	**2714** (p)	**2620** (d)	**2620** (d)
0.003	0.05	1559 (s)	1559 (s)	1559 (s)	1559 (s)	1559 (s)	1559 (s)
0.004	0.05	554 (s)	554 (s)	554 (s)	554 (s)	554 (s)	554 (s)
0.005	0.1	444 (s)	444 (s)	444 (s)	444 (s)	444 (s)	444 (s)
0.01	0.25	113 (s)	113 (s)	113 (s)	113 (s)	113 (s)	113 (s)
0.01	n/a	5622 (fr)	n/a	**36700** (fr)	**2443** (fp)	5622 (fr)	5622 (fr)
0.02		2677 (fr)		**17762** (fr)	**1223** (fp)	2677 (fr)	2677 (fr)
0.05		889 (fr)		**6372** (fr)	**490** (fp)	889 (fr)	889 (fr)
0.1		246 (fp)		**2466** (fr)	246 (fp)	**237** (ed)	**237** (ed)
0.2		124 (fp)		124 (fp)	124 (fp)	**120** (ed)	**120** (ed)
0.3		84 (fp)		84 (fp)	84 (fp)	**80** (ed)	**80** (ed)
0.4		63 (fp)		63 (fp)	63 (fp)	**61** (ed)	**61** (ed)
0.5		51 (fp)		**75** (fr)	51 (fp)	**49** (ed)	**49** (ed)

Table D.8: Predicted Number of Traces for Inclusion in the Mined Model of Noise from $\mathcal{O}_1$ (Top) or $\mathcal{O}_2$ (Bottom), varying RTB and PO Parameters. Determining Factors (s) $\mathrm{DM}_{be} > \mathrm{DM}_{de}$, (r) $|\,\mathrm{DM}_{ic} - \mathrm{DM}_{ac}\,| < RTB$, (p) $N(ic) > \mathrm{PO}$, (d) $\mathrm{DM}_{ic} > \mathrm{DT}$, (fr) $|\,\mathrm{DM}_{fo} - \mathrm{DM}_{co}\,| < RTB$, (fp) $N(fo) > \mathrm{PO}$, (ed) $\mathrm{DM}_{eo} > \mathrm{DT}$.

κ	minimum distance	defaults	DT 0.5	0.8	0.95	0.99
0.001	0.05	6676 (r)	6675 (r)	6675 (r)	**13058** (d)	**28289** (s)
0.002	0.05	2714 (p)	2714 (p)	2714 (p)	**5025** (s)	**5025** (s)
0.003	0.05	1559 (s)	1559 (s)	1559 (s)	1559 (s)	1559 (s)
0.004	0.05	554 (s)	554 (s)	554 (s)	554 (s)	554 (s)
0.005	0.1	444 (s)	444 (s)	444 (s)	444 (s)	444 (s)
0.01	0.25	113 (s)	113 (s)	113 (s)	113 (s)	113 (s)
0.01	n/a	5622 (fr)	5622 (fr)	5622 (fr)	**5878** (ed)	**37775** (ed)
0.02		2677 (fr)	2677 (fr)	2677 (fr)	**2940** (ed)	**18892** (ed)
0.05		889 (fr)	889 (fr)	889 (fr)	**1178** (ed)	**7561** (ed)
0.1		246 (fp)	246 (fp)	246 (fp)	**590** (ed)	**3784** (ed)
0.2		124 (fp)	124 (fp)	124 (fp)	**297** (ed)	**1896** (ed)
0.3		84 (fp)	84 (fp)	84 (fp)	**199** (ed)	**1266** (ed)
0.4		63 (fp)	63 (fp)	63 (fp)	**150** (ed)	**952** (ed)
0.5		51 (fp)	51 (fp)	51 (fp)	**121** (ed)	**763** (ed)

Table D.9: Predicted Numbers of Traces for Inclusion in the Mined Model of Noise from $\mathcal{O}_1$ (Top) or $\mathcal{O}_2$ (Bottom), varying DT parameter. Determining Factors (s) $\mathrm{DM}_{be} > \mathrm{DM}_{de}$, (r) $|\,\mathrm{DM}_{ic} - \mathrm{DM}_{ac}\,| < RTB$, (p) $N(ic) >$ PO, (d) $\mathrm{DM}_{ic} >$ DT, (fr) $|\,\mathrm{DM}_{fo} - \mathrm{DM}_{co}\,| < RTB$, (fp) $N(fo) >$ PO, (ed) $\mathrm{DM}_{eo} >$ DT.

LIST OF REFERENCES

[1] *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, Paris*. IEEE, 2011.

[2] N. Abe and M. K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. In *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, pages 52–66, San Mateo, CA, USA, 1990.

[3] R. Accorsi and T. Stocker. Discovering workflow changes with time-based trace clustering. In K. Aberer, E. Damiani, and T. S. Dillon, editors, *SIMPDA*, volume 116 of *LNBIP*, pages 154–168. Springer, 2011.

[4] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, and W. M. P. van der Aalst. Alignment based precision checking. In Rosa and Soffer [121], pages 137–149.

[5] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst. Towards robust conformance checking. In M. zur Muehlen and J. Su, editors, *BPM Workshops*, volume 66 of *LNBIP*, pages 122–133. Springer, 2010.

[6] A. Adriansyah, B. F. van Dongen, and W. M. P. van der Aalst. Conformance checking using cost-based fitness analysis. In *EDOC*, pages 55–64. IEEE Computer Society, 2011.

[7] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In H.-J. Schek, F. Saltor, I. Ramos, and G. Alonso, editors, *EDBT*, volume 1377 of *LNCS*, pages 469–483. Springer, 1998.

[8] F. Aiolli, A. Burattin, and A. Sperduti. A business process metric based on the Alpha algorithm relations. In Daniel et al. [43], pages 141–146.

[9] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFst: A general and efficient weighted finite-state transducer library. In J. Holub and J. Zdárek, editors, *CIAA*, volume 4783 of *LNCS*, pages 11–23. Springer, 2007.

[10] L. Allison. KL-distance between Gaussians. `http://www.allisons.org/ll/MML/KL/Normal/` (retrieved 8/1/2013).

[11] G. Alonso, P. Dadam, and M. Rosemann, editors. *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Proceedings*, volume 4714 of *LNCS*. Springer, 2007.

[12] A. K. A. de Medeiros, W. M. P. van der Aalst, and A. J. M. M. Weijters. Quantifying process equivalence based on observed behavior. *Data and Knowledge Engineering*, 64(1):55–74, 2008.

[13] D. Angluin. Identifying languages from stochastic examples. Technical Report YALEU/DCS/TR614, Yale University, 1988.

[14] D. Angluin. Computational learning theory: Survey and selected bibliography. In *STOC*, pages 351–369. ACM, 1992.

[15] J. Bae, L. Liu, J. Caverlee, and W. B. Rouse. Process mining, discovery, and integration using distance measures. In *Proceedings of the 2006 IEEE International Conference on Web Services (ICWS)*, pages 479–486, Chicago, 2006.

[16] J. Bae, L. Liu, J. Caverlee, L.-J. Zhang, and H. Bae. Development of distance measures for process mining, discovery, and integration. *International Journal of Web Services Research*, 4(4):1–17, 2007.

[17] L. Baresi and M. Pezzè. On formalizing UML with high-level Petri Nets. In G. Agha, F. de Cindio, and G. Rozenberg, editors, *Concurrent Object-Oriented Programming and Petri Nets*, volume 2001 of *LNCS*, pages 276–304. Springer, 2001.

[18] M. Becker and R. Laue. Analysing differences between business process similarity measures. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *BPM Workshops (2)*, volume 100 of *LNBIP*, pages 39–49. Springer, 2011.

[19] E. Bellodi, F. Riguzzi, and E. Lamma. Probabilistic declarative process mining. In Y. Bi and M.-A. Williams, editors, *KSEM*, volume 6291 of *LNCS*, pages 292–303. Springer, 2010.

[20] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Process mining based on regions of languages. In Alonso et al. [11], pages 375–383.

[21] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35:99–109, 1943.

[22] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[23] B. Bordbar and P. Weber. Automated prevention of failure in complex and large systems: Fighting fire with fire. *Accepted for publication in International Journal of Informatics Society (IJIS)*, 5:(to appear), 2013.

[24] R. P. J. C. Bose and W. M. P. van der Aalst. Context aware trace clustering: Towards improving process mining results. In *SDM*, pages 401–412. SIAM, 2009.

[25] R. P. J. C. Bose and W. M. P. van der Aalst. Trace clustering based on conserved patterns: Towards achieving better process models. In Rinderle-Ma et al. [119], pages 170–181.

[26] R. P. J. C. Bose, W. M. P. van der Aalst, I. Zliobaite, and M. Pechenizkiy. Handling concept drift in process mining. In H. Mouratidis and C. Rolland, editors, *CAiSE*, volume 6741 of *LNCS*, pages 391–405. Springer, 2011.

[27] R. P. J. C. Bose, H. M. W. Verbeek, and W. M. P. van der Aalst. Discovering hierarchical process models using ProM. In S. Nurcan, editor, *CAiSE Forum*, volume 734 of *CEUR Workshop Proceedings*, pages 33–40. CEUR-WS.org, 2011.

[28] A. Burattin and A. Sperduti. Automatic determination of parameters' values for Heuristics Miner++. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[29] A. Burattin and A. Sperduti. Heuristics Miner for time intervals. In *Proceedings of ESANN 2010*, Bruges, Belgium, 2010.

[30] G. C. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Kluwer Academic Publishers, Norwell, USA, 1997.

[31] E. Byres and J. Lowe. Myths and facts behind cyber security risks for industrial control systems. *Engineering Technology*, 7(10):48–50, 2004-2005.

[32] T. Calders, C. W. Günther, M. Pechenizkiy, and A. Rozinat. Using minimum description length for process mining. In S. Y. Shin and S. Ossowski, editors, *SAC*, pages 1451–1455. ACM, 2009.

[33] R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R. C. Carrasco and J. Oncina, editors, *ICGI*, volume 862 of *LNCS*, pages 139–152. Springer, 1994.

[34] A. Cedilnik, K. Krošmelj, and A. Blejec. The distribution of the ratio of jointly Normal variables. *Metodološki zvezki – Advances in Methodology and Statistics*, 1(1):99–108, 2004.

[35] J. Claes and G. Poels. Process mining and the ProM framework: an exploratory survey. In Rosa and Soffer [121], pages 187–198.

[36] A. Clark and F. Thollard. PAC-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5:473–497, 2004.

[37] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–77, 2003.

[38] J. E. Cook, Z. Du, C. Liu, and A. L. Wolf. Discovering models of behavior for concurrent workflows. *Computers in Industry*, 53(3):297–319, 2004.

[39] J. E. Cook and A. L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Softw. Eng. Methodol.*, 7(3):215–249, 1998.

[40] J. E. Cook and A. L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. *ACM Transactions on Softw. Eng. Methodol.*, 8(2):147–176, 1999.

[41] C. Cortes, M. Mohri, and A. Rastogi. On the computation of some standard distances between probabilistic automata. In O. H. Ibarra and H.-C. Yen, editors, *CIAA*, volume 4094 of *LNCS*, pages 137–149. Springer, 2006.

[42] C. Cortes, M. Mohri, A. Rastogi, and M. Riley. Efficient computation of the relative entropy of probabilistic automata. In J. R. Correa, A. Hevia, and M. A. Kiwi, editors, *LATIN*, volume 3887 of *LNCS*, pages 323–336. Springer, 2006.

[43] F. Daniel, K. Barkaoui, and S. Dustdar, editors. *BPM Workshops, BPM 2011 International Workshops, Clermont-Ferrand, France, 2011, Revised Selected Papers, Part I*, volume 99 of *LNBIP*. Springer, 2012.

[44] A. Datta. Automating the discovery of AS-IS business process models: probabilistic and algorithmic approaches. *Information Systems Research*, 9(3):275–301, 1998.

[45] F. de Lima Bezerra, J. Wainer, and W. M. P. van der Aalst. Anomaly detection using process mining. In T. A. Halpin, J. Krogstie, S. Nurcan, E. Proper, R. Schmidt, P. Soffer, and R. Ukor, editors, *BMMDS/EMMSAD*, volume 29 of *LNBIP*, pages 149–161. Springer, 2009.

[46] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, 14(2):245–304, 2007.

[47] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst. Genetic process mining: A basic approach and its challenges. In volume 2812 of *LNCS*, pages 203–215, Heidelberg, 2005.

[48] A. K. A. de Medeiros, B. F. van Dongen, W. M. P. van der Aalst, and A. J. M. M. Weijters. Process mining: Extending the Alpha algorithm to mine short loops, *BETA Working Paper 113*, Eindhoven University of Technology, 2004.

[49] A. K. A. de Medeiros, A. J. M. M. Weijters, and W. M. P. van der Aalst. Using Genetic Algorithms to mine process models: Representation, operators and results, *BETA Working Paper 124*, Eindhoven University of Technology, 2004.

[50] A. K. A. de Medeiros. *Genetic Process Mining*. PhD thesis, Eindhoven : Technische Universiteit Eindhoven, 2006.

[51] A. K. A. de Medeiros, A. Guzzo, G. Greco, W. M. P. van der Aalst, A. J. M. M. Weijters, B. F. van Dongen, and D. Saccà. Process mining based on clustering: A quest for precision. In ter Hofstede et al. [142], pages 17–29.

[52] J. de Weerdt, M. de Backer, J. Vanthienen, and B. Baesens. A robust F-measure for evaluating discovered process models. pages 148–155, Paris, 2011.

[53] R. M. Dijkman, M. Dumas, B. F. van Dongen, R. Käärik, and J. Mendling. Similarity of business process models: Metrics and evaluation. *Information Systems*, 36(2):498–516, 2011.

[54] B. F. van Dongen. Multi-phase process mining: aggregating instance graphs into EPCs and Petri Nets. In *In PNCWB 2005 workshop*, pages 35–58, 2005.

[55] B. F. van Dongen, N. Busi, and G. M. Pinna. An iterative algorithm for applying the theory of regions in process mining, *BETA Working Paper 195*, Eindhoven University of Technology, 2007.

[56] M. H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice-Hall, 2002.

[57] P. Dupont, F. Denis, and Y. Esposito. Links between probabilistic automata and Hidden Markov Models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, 38(9):1349–71, 2005.

[58] W. H. Dutton and G. Blank. Next generation users: the internet in Britain. *Oxford Internet Survey*, 2011. Oxford Internet Institute, University of Oxford.

[59] D. Fahland and W. M. P. van der Aalst. Repairing process models to reflect reality. In A. P. Barros, A. Gal, and E. Kindler, editors, *BPM*, volume 7481 of *LNCS*, pages 229–245. Springer, 2012.

[60] D. Fahland, M. de Leoni, B. F. van Dongen, and W. M. P. van der Aalst. Conformance checking of interacting processes with overlapping instances. In Rinderle-Ma et al. [120], pages 345–361.

[61] J. P. Farwell and R. Rohozinski. Stuxnet and the future of cyber war. *Survival*, 53(1):23–40, 2011.

[62] N. Ferraro, L. Palopoli, S. Panni, and S. E. Rombo. Asymmetric comparison and querying of biological networks. *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, 8(4):876–889, 2011.

[63] D. R. Ferreira and D. Gillblad. Discovering process models from unlabelled event logs. In U. Dayal, J. Eder, J. Koehler, and H. A. Reijers, editors, *BPM*, volume 5701 of *LNCS*, pages 143–158. Springer, 2009.

[64] F. Folino, G. Greco, A. Guzzo, and L. Pontieri. Discovering expressive process models from noised log data. In B. C. Desai, D. Saccà, and S. Greco, editors, *IDEAS*, ACM International Conference Proceeding Series, pages 162–172. ACM, 2009.

[65] W. Gaaloul, K. Gaaloul, S. Bhiri, A. Haller, and M. Hauswirth. Log-based transactional workflow mining. *Distributed and Parallel Databases*, 25(3):193–240, 2009.

[66] D. Gao and Q. Liu. An improved simulated annealing algorithm for process mining. In *2009 13th International Conference on Computer Supported Cooperative Work in Design*, pages 474–9, Piscataway, NJ, USA, 2009.

[67] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens. Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, 10:1305–1340, 2009.

[68] S. Goedertier, J. de Weerdt, D. Martens, J. Vanthienen, and B. Baesens. Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing*, 11(2):1697–1710, 2011.

[69] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[70] G. Greco, A. Guzzo, and L. Pontieri. Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1010–1027, 2006.

[71] G. Greco, A. Guzzo, and L. Pontieri. Mining taxonomies of process models. *Data and Knowledge Engineering*, 67(1):74–102, 2008.

[72] G. Greco, A. Guzzo, G. Manco, L. Pontieri, and D. Saccà. Mining constrained graphs: The case of workflow systems. In J.-F. Boulicaut, L. de Raedt, and H. Mannila, editors, *Constraint-Based Mining and Inductive Databases*, volume 3848 of *LNCS*, pages 155–171. Springer, 2004.

[73] C. W. Günther and W. M. P. van der Aalst. Fuzzy mining – adaptive process simplification based on multi-perspective metrics. In Alonso et al. [11], pages 328–343.

[74] C. W. Günther, A. Rozinat, and W. M. P. van der Aalst. Activity mining by global trace segmentation. In Rinderle-Ma et al. [119], pages 128–139.

[75] D. Haussler. Probably approximately correct learning. In H. E. Shrobe, T. G. Dietterich, and W. R. Swartout, editors, *AAAI*, pages 1101–1108. AAAI Press / The MIT Press, 1990.

[76] M. Havey. *Essential Business Process Modeling.* O'Reilly Media, Inc., 2005.

[77] J. Herbst and D. Karagiannis. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 9(2):67–92, 2000.

[78] J. Herbst and D. Karagiannis. Workflow mining with InWoLvE. *Computers in Industry*, 53(3):245–264, 2004.

[79] J. Herbst. A machine learning approach to workflow management. In R. L. de Mántaras and E. Plaza, editors, *ECML*, volume 1810 of *LNCS*, pages 183–194. Springer, 2000.

[80] J. Herbst and D. Karagiannis. Integrating machine learning and workflow management to support acquisition and adaption of workflow models. In *DEXA Workshop*, pages 745–752, 1998.

[81] S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri Nets. In W. M. P. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Business Process Management*, volume 3649, pages 220–235, 2005.

[82] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[83] S. Jacquemont, F. Jacquenet, and M. Sebban. Mining probabilistic automata: A statistical view of sequential pattern mining. *Machine Learning*, 75(1):91–127, 2009.

[84] M. Jans, J. M. E. M. van der Werf, N. Lybaert, and K. Vanhoof. A business process mining application for internal transaction fraud mitigation. *Expert Systems with Applications*, 38(10), 2011.

[85] S. Jianchun and Y. Dongqing. Process mining: Algorithm for s-coverable workflow nets. In *WKDD*, pages 239–244. IEEE Computer Society, 2009.

[86] M. J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.

[87] G. Keller, M. Nüttgens, and A. W. Scheer. Semantische Prozessmodellierung auf der Grundlage Ereignisgesteuerter Prozessketten (EPK). 1992. *Institut für Wirtschaftsinformatik, Universität des Saarlandes: Saarbrücken*, 2009.

[88] R. Khalaf, N. Mukhi, F. Curbera, and S. Weerawarana. The business process execution language for web services. In *Process-Aware Information Systems*. Wiley, 2005.

[89] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[90] C. B. Lassen and W. M. P. van der Aalst. Complexity metrics for workflow nets. *Information and Software Technology*, 51(3):610–626, 2009.

[91] J. Li, R. P. J. C. Bose, and W. M. P. van der Aalst. Mining context-dependent and interactive business process maps using execution patterns. In volume 66 of *LNBIP*, pages 109–121, Hoboken, NJ, United States, 2011.

[92] J. Li, D. Liu, and B. Yang. Process mining: Extending α-algorithm to mine duplicate tasks in process logs. In K. C.-C. Chang, W. Wang, L. Chen, C. A. Ellis, C.-H. Hsu, A. C. Tsoi, and H. Wang, editors, *APWeb/WAIM Workshops*, volume 4537 of *LNCS*, pages 396–407. Springer, 2007.

[93] J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[94] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[95] N. Lohmann, E. Verbeek, and R. M. Dijkman. Petri Net transformations for business processes – a survey. *T. Petri Nets and Other Models of Concurrency*, 2:46–63, 2009.

[96] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst. Efficient discovery of understandable declarative process models from event logs. In J. Ralyté, X. Franch, S. Brinkkemper, and S. Wrycza, editors, *CAiSE*, volume 7328 of *LNCS*, pages 270–285. Springer, 2012.

[97] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst. User-guided discovery of declarative process models. In *CIDM* [1], pages 192–199.

[98] G. K. Manacher. Production and stabilization of real-time task schedules. *Journal of the ACM*, 14(3):439–465, 1967.

[99] R. S. Mans, H. Schonenberg, M. Song, W. M. P. van der Aalst, and P. J. M. Bakker. Application of process mining in healthcare – A case study in a Dutch hospital. In A. L. N. Fred, J. Filipe, and H. Gamboa, editors, *BIOSTEC (Selected Papers)*, volume 25 of *Communications in Computer and Information Science*, pages 425–438. Springer, 2008.

[100] G. Marsaglia. Ratios of Normal variables. *Journal of Statistical Software*, 16(4):1–10, 2006.

[101] G. Marsaglia. Ratios of Normal variables and ratios of sums of uniform variables. *Journal of the American Statistical Association*, 60:193–204, 1965.

[102] M. Minor, A. Tartakovski, and R. Bergmann. Representation and structure-based similarity assessment for agile workflows. In R. Weber and M. M. Richter, editors, *ICCBR*, volume 4626 of *LNCS*, pages 224–238. Springer, 2007.

[103] T. M. Mitchell. The discipline of machine learning. Technical Report CMU-ML-06-108, Carnegie Mellon University, Machine Learning Department, 2006.

[104] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[105] P. M. Esfahani, M. Vrakopoulou, K. Margellos, J. Lygeros, and G. Andersson. Cyber attack in a two-area power system: Impact identification using reachability. In *2010 American Control Conference (ACC 2010)*, pages 962–7, Piscataway, NJ, USA, 2010.

[106] J. Muñoz-Gama and J. Carmona. A fresh look at precision in process conformance. In R. Hull, J. Mendling, and S. Tai, editors, *BPM*, volume 6336 of *LNCS*, pages 211–226. Springer, 2010.

[107] T. Murata. Petri Nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[108] L. Măruşter. *A Machine Learning Approach to Understand Business Processes*. PhD thesis, Eindhoven : Technische Universiteit Eindhoven, 2003.

[109] OMG. Business Process Model and Notation (BPMN), 2009.

[110] K. Pearson On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophical Magazine Series 5*, 50(302), pages 157–175, 1900.

[111] W. Penny and S. Roberts. Variational Bayes for generalised autoregressive models. Technical Report PARG-00-12, Department of Engineering Science, University of Oxford, 2000.

[112] J. L. Peterson. Petri Nets. *ACM Computing Surveys*, 9(3):223–252, 1977.

[113] S. S. Pinter and M. Golani. Discovering workflow models from activities' lifespans. *Computers in Industry*, 53(3):283–96, 2004.

[114] R. M. Podorozhny, A. H. H. Ngu, and D. Georgakopoulos. Business process learning for real time enterprises. In C. Bussler, M. Castellanos, U. Dayal, and S. B. Navathe, editors, *BIRTE*, volume 4365 of *LNCS*, pages 118–132. Springer, 2006.

[115] P. Puschner and C. Koza. Calculating the maximum execution time of real-time programs. *Real-Time Systems*, 1:159–176, 1989.

[116] I. Raedts, M. Petkovic, Y. S. Usenko, J. M. E. M. van der Werf, J. F. Groote, and L. J. Somers. Transformation of BPMN models for behaviour analysis. In J. C. Augusto, J. Barjis, and U. Ultes-Nitsche, editors, *MSVVEIS*, pages 126–137. INSTICC PRESS, 2007.

[117] C. Ren, L. Wen, J. Dong, H. Ding, W. Wang, and M. Qiu. A novel approach for process mining based on event types. In *IEEE SCC*, pages 721–722. IEEE Computer Society, 2007.

[118] S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems – a survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.

[119] S. Rinderle-Ma, S. W. Sadiq, and F. Leymann, editors. *BPM Workshops, BPM 2009 International Workshops, Ulm, Germany, 2009. Revised Papers*, volume 43 of *LNBIP*. Springer, 2010.

[120] S. Rinderle-Ma, F. Toumani, and K. Wolf, editors. *Business Process Management, 9th International Conference, BPM 2011, Clermont-Ferrand, France, 2011, Proceedings*, volume 6896 of *LNCS*. Springer, 2011.

[121] M. La Rosa and P. Soffer, editors. *BPM Workshops, BPM 2012 International Workshops, Tallinn, Estonia, 2012. Revised Papers*, volume 132 of *LNBIP*. Springer, 2013.

[122] A. Rozinat, A. K. A. de Medeiros, C. W. Günther, A. J. M. M. Weijters, and W. M. P. van der Aalst. Towards an evaluation framework for process mining algorithms. *BPM Center Report BPM-07-06*, 2007.

[123] A. Rozinat, R. S. Mans, M. Song, and W. M. P. van der Aalst. Discovering simulation models. *Information Systems*, 34(3):305–327, 2009.

[124] A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.

[125] A. Rozinat, M. T. Wynn, W. M. P. van der Aalst, A. H. M. ter Hofstede, and C. J. Fidge. Workflow simulation for operational decision support using YAWL and ProM. *BPM Center Report BPM-08-04*, 2008.

[126] A. Rozinat, I. S. M. de Jong, C. W. Günther, and W. M. P. van der Aalst. Process mining applied to the test process of wafer scanners in ASML. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 39(4):474–479, 2009.

[127] A. Rozinat and W. M. P. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In C. Bussler and A. Haller, editors, *BPM Workshops*, volume 3812, pages 163–176, 2005.

[128] A. Rozinat and W. M. P. van der Aalst. Decision mining in ProM. In S. Dustdar, J. L. Fiadeiro, and A. P. Sheth, editors, *Business Process Management*, volume 4102 of *LNCS*, pages 420–425. Springer, 2006.

[129] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004.

[130] A. R. Runnalls. Kullback-Leibler approach to Gaussian mixture reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 43(3):989–999, 2007.

[131] D. Ruta and B. Majeed. Business process forecasting in telecom industry. In *2011 IEEE GCC Conference and Exhibition, GCC 2011*, pages 389–392, Dubai, UAE, 2011.

[132] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[133] G. Schimm. Mining exact models of concurrent workflows. *Computers in Industry*, 53(3):265–281, 2004.

[134] H. Schonenberg, N. Sidorova, W. M. P. van der Aalst, and K. M. van Hee. History-dependent stochastic Petri Nets. In A. Pnueli, I. Virbitskaite, and A. Voronkov, editors, *Ershov Memorial Conference*, volume 5947 of *LNCS*, pages 366–379. Springer, 2009.

[135] H. Schonenberg, B. Weber, B. F. van Dongen, and W. M. P. van der Aalst. Supporting flexible processes through recommendations based on history. In M. Dumas, M. Reichert, and M.-C. Shan, editors, *BPM*, volume 5240 of *LNCS*, pages 51–66. Springer, 2008.

[136] G. W. Snedecor and W. G. Cochran. *Statistical Methods*. Iowa State University Press, Ames IO, 8th edition, 1989.

[137] M. Song, C. W. Günther, and W. M. P. van der Aalst. Trace clustering in process mining. In D. Ardagna, M. Mecella, and J. Yang, editors, *BPM Workshops*, volume 17 of *LNBIP*, pages 109–120. Springer, 2008.

[138] W. Song, S. Liu, and Q. Liu. Business process mining based on simulated annealing. In *Proceedings of The 9th International Conference for Young Computer Scientists (ICYCS)*, pages 725–730, Washington, DC, USA, 2008. IEEE Computer Society.

[139] A. Staikopoulos and B. Bordbar. A metamodel refinement approach for bridging technical spaces, a case study. In *Proceedings of the 4th MoDELS Workshop in Software Model Engineering (WiSME 2005)*, 2005.

[140] K. D. Swenson, N. Palmer, J. P. Ukelson, T. Shepherd, and J. T. Matthias. *Mastering the Unpredictable*. Meghan-Kiffer Press, New York, 2010.

[141] P. Taylor, M. Leida, and B. Majeed. Case study in process mining in a multinational enterprise. In K. Aberer, E. Damiani, and T. Dillon, editors, *Data-Driven*

Process Discovery and Analysis, volume 116 of *LNBIP*, pages 134–153. Springer Berlin Heidelberg, 2012.

[142] A. H. M. ter Hofstede, B. Benatallah, and H.-Y. Paik, editors. *BPM Workshops, BPM 2007 International Workshops, Brisbane, 2007, Revised Selected Papers*, volume 4928 of *LNCS*. Springer, 2008.

[143] F. Thollard, P. Dupont, and F. de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In P. Langley, editor, *ICML*, pages 975–982. Morgan Kaufmann, 2000.

[144] A. Tiwari, C. J. Turner, and B. Majeed. A review of business process mining: state-of-the-art and future trends. *Business Process Management Journal*, 14(1):5–22, 2008.

[145] C. J. Turner, A. Tiwari, and J. Mehnen. A genetic programming approach to business process mining. In C. Ryan and M. Keijzer, editors, *GECCO*, pages 1307–1314. ACM, 2008.

[146] R. Uba, M. Dumas, L. García-Bañuelos, and M. La Rosa. Clone detection in repositories of business process models. In Rinderle-Ma et al. [120], pages 248–264.

[147] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[148] W. M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, Heidelberg Dordrecht London New York, 2011.

[149] W. M. P. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, R. P. J. C. Bose, P. van den Brand, R. Brandtjen, J. C. A. M. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B. F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. W. Günther, A. Guzzo, P. Harmon, A. H. M. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F. M. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. M. Nezhad, M. zur Muehlen, J. Muñoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. S. Pérez, R. S. Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. D. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M. T. Wynn. Process mining manifesto. In Daniel et al. [43], pages 169–194.

[150] W. M. P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[151] W. M. P. van der Aalst, A. K. A. de Medeiros, and A. J. M. M. Weijters. Genetic process mining. In *Applications and Theory of Petri Nets 2005. 26th International Conference, ICATPN 2005, Proceedings*, volume 3536 of *LNCS*, pages 48–69, Berlin, 2005.

[152] W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. de Medeiros, M. Song, and H. M. W. Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007.

[153] W. M. P. van der Aalst, H. A. Reijers, and M. Song. Discovering social networks from event logs. *Computer Supported Cooperative Work*, 14(6):549–593, 2005.

[154] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.

[155] W. M. P. van der Aalst and A. J. M. M. Weijters. Process mining: a research agenda. *Computers in Industry*, 53(3):231–244, 2004.

[156] W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[157] W. M. P. van der Aalst. The application of Petri Nets to workflow management. *Journal of Circuits, Systems, and Computers*, 8(1):21–66, 1998.

[158] W. M. P. van der Aalst. Trends in business process analysis – from verification to process mining. In J. Cardoso, J. Cordeiro, and J. Filipe, editors, *ICEIS (1)*, pages 5–9, 2007.

[159] W. M. P. van der Aalst. On the representational bias in process mining. In S. Reddy and S. Tata, editors, *WETICE*, pages 2–7. IEEE Computer Society, 2011.

[160] W. M. P. van der Aalst. Decomposing process mining problems using passages. In S. Haddad and L. Pomello, editors, *Petri Nets*, volume 7347 of *LNCS*, pages 72–91. Springer, 2012.

[161] W. M. P. van der Aalst, J. Desel, and E. Kindler. On the semantics of EPCs: A vicious circle. In M. Nüttgens and F. J. Rump, editors, *EPK*, pages 71–79. GI-Arbeitskreis Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, 2002.

[162] W. M. P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H. M. W. Verbeek. Choreography Conformance Checking: An Approach based on BPEL and Petri Nets. In F. Leymann, W. Reisig, S. R. Thatte, and W. M. P. van der Aalst, editors, *The Role of Business Processes in Service Oriented Architectures*, volume 06291 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[163] W. M. P. van der Aalst and C. W. Günther. Finding structure in unstructured processes: The case for process mining. In T. Basten, G. Juhás, and S. K. Shukla, editors, *ACSD*, pages 3–12. IEEE Computer Society, 2007.

[164] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and System Modeling*, 9(1):87–111, 2010.

[165] W. M. P. van der Aalst, M. H. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011.

[166] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*, 30(4):245–275, 2005.

[167] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.

[168] W. M. P. van der Aalst, A. K. A. de Medeiros, and A. J. M. M. Weijters. Process equivalence: comparing two process models based on observed behavior. In *Business Process Management, 4th International Conference, BPM 2006, Proceedings*, volume 4102 of *LNCS*, pages 129–44, Berlin, 2006.

[169] W. M. P. van der Aalst, V. Rubin, H. M. W. Verbeek, B. F. van Dongen, E. Kindler, and C. W. Günther. Process mining: A two-step approach to balance between underfitting and overfitting. *Software and Systems Modeling*, 9(1):87–111, 2010.

[170] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The ProM framework: A new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *ICATPN*, volume 3536 of *LNCS*, pages 444–454. Springer, 2005.

[171] B. F. van Dongen and W. M. P. van der Aalst. Multi-phase process mining: building instance graphs. In *Conceptual Modeling – ER 2004. 23rd International Conference on Conceptual Modeling, Proceedings*, volume 3288 of *LNCS*, pages 362–76, Berlin, 2004.

[172] B. F. van Dongen and A. Adriansyah. Process mining: Fuzzy clustering and performance visualization. In Rinderle-Ma et al. [119], pages 158–169.

[173] B. F. van Dongen, R. M. Dijkman, and J. Mendling. Measuring similarity between business process models. In Z. Bellahsene and M. Léonard, editors, *CAiSE*, volume 5074 of *LNCS*, pages 450–464. Springer, 2008.

[174] B. F. van Dongen, J. Mendling, and W. M. P. van der Aalst. Structural patterns for soundness of business process models. In *EDOC*, pages 116–128. IEEE Computer Society, 2006.

[175] K. M. van Hee, Z. Liu, and N. Sidorova. Is my event log complete? – A probabilistic approach to process mining. In *RCIS*, pages 1–7. IEEE, 2011.

[176] J. Vanhatalo, H. Völzer, and J. Koehler. The refined process structure tree. *Data and Knowledge Engineering*, 68(9):793–818, 2009.

[177] H. M. W. Verbeek and B. F. van Dongen. Translating labelled P/T nets into EPCs for sake of communication. *BETA Working Paper 194*, Eindhoven University of Technology, 2007.

[178] H. M. W. Verbeek, J. C. A. M. Buijs, B. F. Van Dongen and W. M. P. van der Aalst. XES, XESame, and ProM 6. In P. Soffer and E. Proper, editors, *Information Systems Evolution (CAiSE Forum 2010)*, volume 72 of *LNBIP*, pages 60–75. Springer, 2011.

[179] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 27(7):1013–25, 2005.

[180] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1026–39, 2005.

[181] A. Wang, W. Zhao, C. Chen, and H. Wu. A GP process mining approach from a structural perspective. In *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence, AICI 2009*, pages 121–30, Berlin, 2009.

[182] J. Wang, T. He, L. Wen, N. Wu, A. H. M. ter Hofstede, and J. Su. A behavioral similarity measure between labeled Petri Nets based on principal transition sequences – (short paper). In R. Meersman, T. S. Dillon, and P. Herrero, editors, *OTM Conferences (1)*, volume 6426 of *LNCS*, pages 394–401. Springer, 2010.

[183] B. Weber, S. W. Sadiq, and M. Reichert. Beyond rigidity – dynamic process lifecycle support. *Computer Science – R&D*, 23(2):47–65, 2009.

[184] P. Weber, B. Bordbar, and P. Tiňo. A framework for the analysis of process mining algorithms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(2):303–317, 2013.

[185] P. Weber, B. Bordbar, and P. Tiňo. A principled approach to the analysis of process mining algorithms. In H. Yin, W. Wang, and V. J. Rayward-Smith, editors, *IDEAL*, volume 6936 of *LNCS*, pages 474–481. Springer, 2011.

[186] P. Weber, B. Bordbar, and P. Tiňo. Real-time detection of process change using process mining. In A. V. Jones, editor, *ICCSW*, volume DTR11-9 of *Department of Computing Technical Report*, pages 108–114. Imperial College London, 2011.

[187] P. Weber, B. Bordbar, P. Tiňo, and B. Majeed. A framework for comparing process mining algorithms. In *2011 IEEE GCC Conference and Exhibition (GCC)*, pages 625–628, Dubai, UAE, 2011.

[188] P. Weber, P. Tiňo, and B. Bordbar. Process mining in non-stationary environments. In *Proceedings of ESANN 2012*, Bruges, Belgium, 2012.

[189] P. Weber, B. Bordbar, and P. Tiňo. A principled approach to mining from noisy logs using Heuristics Miner. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 119–26, Piscataway, NJ, USA, 2013.

[190] P. Weber, P. N. Taylor, B. Majeed, and B. Bordbar. Comparing complex business process models. In *IEEE International Conference on Industrial Engineering and Engineering Management – IEEM 2012*, page (to appear), 2012.

[191] J. de Weerdt, M. de Backer, J. Vanthienen, and B. Baesens. A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs. *Information Systems*, 37(7):654–676, 2012.

[192] M. Weidlich, A. Polyvyanyy, N. Desai, and J. Mendling. Process compliance measurement based on behavioural profiles. In B. Pernici, editor, *CAiSE*, volume 6051 of *LNCS*, pages 499–514. Springer, 2010.

[193] A. J. M. M. Weijters and J. T. S. Ribeiro. Flexible Heuristics Miner (FHM). In *CIDM* [1], pages 310–317.

[194] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. A. de Medeiros. Process mining with the HeuristicsMiner algorithm. *BETA Working Paper Series 166*, Eindhoven University of Technology, 2006.

[195] L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.

[196] Workflow Patterns Initiative. Workflow patterns, 2007. `http://www.workflowpatterns.com/` (accessed 30/03/2010).

[197] H. Yang, A. H. M. ter Hofstede, B. F. van Dongen, M. T. Wynn, and J. Wang. On global completeness of event logs. *BPM Center Report BPM-10-09*, 2010.

[198] B.-J. Yoon, X. Qian, and S. M. E. Sahraeian. Comparative analysis of biological networks: Hidden Markov Model and Markov chain-based approach. *IEEE Signal Processing Magazine*, 29(1):22–34, 2012.

[199] D. Yue, X. Wu, H. Wang, and J. Bai. A review of process mining algorithms. In *Business Management and Electronic Information (BMEI), 2011 International Conference on*, volume 5, pages 181–185, 2011.

[200] H. Zha, J. Wang, L. Wen, and C. Wang. A label-free similarity measure between workflow nets. In M. Kirchberg, P. C. K. Hung, B. Carminati, C.-H. Chi, R.

Kanagasabai, E. D. Valle, K.-C. Lan, and L.-J. Chen, editors, *APSCC*, pages 463–469. IEEE, 2009.

[201] H. Zha, J. Wang, L. Wen, C. Wang, and J. Sun. A workflow net similarity measure based on transition adjacency relations. *Computers in Industry*, 61(5):463–471, 2010.