Process Mining with Labelled Stochastic Nets

Adam Trent Burke

BSc (Hons)

Submitted in fulfillment of the requirement for the degree of
Doctor of Philosophy

School of Information Systems
Faculty of Science
Queensland University of Technology
2024

Abstract

In process mining, models of processes are built and analysed using large sets of sequential data. Typically, this data is exported from modern IT systems, and analysed for organisational improvement and optimisation. Process mining has been applied to businesses, hospitals, and governments, in domains from manufacturing to education. The probability of particular behaviours or events is important in understanding the behaviour of an organisation, for allocating resources, improving processes, and managing risks. However, techniques for automatic construction and analysis of stochastic process models are few and limited.

This research investigates algorithms and techniques for such analysis using models built using information on event frequency. It contributes solutions to discover stochastic models, calculate metrics and stochastic languages for such models, and investigates novel data sources with stochastic process mining techniques. Process models investigated are labelled stochastic nets that are extensions of Petri Nets and process trees.

The automatic construction of process models is termed process discovery. Novel algorithms for process discovery are introduced, including those which annotate an existing non-stochastic model, and those that require only sequential data as an input. A discovery framework, the Toothpaste Miner, is proposed, using weighted process trees for direct discovery from sequential data in event logs. The formal properties of these Probabilistic Process Trees (PPTs) are leveraged for these algorithms, which are based on reduction rules for identified tree patterns. These are evaluated by experimental comparison against existing stochastic discovery techniques, showing improvements in the breadth of supported logs and in quality metrics. A separate discovery algorithm, the State Snapshot Miner, is developed for discovery of labelled stochastic Petri nets from sequential data on concurrent states. This is applied to a dataset of Qing dynasty civil service records to create career promotion models for use by historians of the period. The application of process mining to data predating modern IT systems is also novel.

The topic of comparing and analysing existing process models is termed process conformance. For stochastic process models, an important problem is calculating the probability a given sequence of activities can happen under a particular process model, termed the trace probability problem. Two novel solutions to this problem are presented, one using PPTs, and another with labelled stochastic Petri nets. Both involve approximation according to an input parameter and complement existing techniques.

The problem of classifying quality metrics in useful conceptual categories on stochastic process models is also investigated. An empirical study is performed on a large dataset of stochastic process models and metrics. The dataset is generated using real-life logs across multiple domains. Three quality dimensions for stochastic models are proposed: Adhesion, Relevance, and Simplicity.

Secondary contributions for log generation and model discovery using genetic mining are also included in this research, and public software implementations are provided for new techniques.

Keywords

process mining, stochastic process mining, process discovery, process conformance, Probabilistic Process Trees, Toothpaste Miner, Stochastic Petri nets, trace probability, process quality dimensions, adhesion, relevance, simplicity

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Meaning	Definition	Page
$\mathcal{A}$	the universe of activities	Section 2.2	20
$\mathcal{B}(C)$	multiset (bag) of set C	Section 2.1.1	19
$\mathcal{B}^{+}(C)$	real-valued multiset of set C	Section 2.1.2	20
det	determinism of Probabilistic Process Trees (PPTs)	Definition 27	55
$\mathcal{L}$	the universe of event logs	Definition 11	24
$\mathcal{L}^{+}$	the universe of play-out event logs	Definition 36	104
$\mathbb{P}(C)$	powerset of set C	Section 2.1	19
$\mathcal{PN}$	set of all place-transition nets	Definition 4	21
$\mathcal{PPT}$	set of all Probabilistic Process Trees	Definition 25	47
$\mathcal{SL}$	set of all State-Snapshot Logs	Definition 43	134
$\mathcal{SN}$	set of all Stochastic Labelled Petri Nets (SLPNs)	Definition 8	22
$\mathcal{WS}$	set of all Weighted Stochastic Finite Automata (WSFAs)	Definition 21	43
$\rightarrow$	sequence operator in Probabilistic Process Trees	Definition 25	47
$\times$	choice operator in Probabilistic Process Trees	Definition 25	47
$\wedge$	concurrency operator in Probabilistic Process Trees	Definition 25	47
$\circlearrowleft_n$	fixed loop operator in Probabilistic Process Trees	Definition 25	47
$\circlearrowleft_p$	probabilistic loop operator in Probabilistic Process Trees	Definition 25	47
$+\!\!+$	concatenation of Weighted Stochastic Finite Automata	Definition 22	44
$\sqcup$	union of Weighted Stochastic Finite Automata	Definition 23	44
$\|\|$	race between Weighted Stochastic Finite Automata	Definition 24	45
$\overset{\sim}{\Longleftrightarrow}$	stochastic preservation on Probabilistic Process Trees	Definition 32	65
$\Rrightarrow_c$	preserving compression rule	Section 4.4.3	61
$\Rightarrow_{fps}$	fitness and precision preserving rule	Section 4.4.3	61
$\Rightarrow_{fs}$	fitness preserving rule	Section 4.4.3	61
$\Rightarrow_s$	simplifying lossy rule	Section 4.4.3	61
Ψ	merge on Probabilistic Process Trees	Section 4.5.1	65
$\cong_c$	similarity on Probabilistic Process Trees	Section 4.5.1	65
$\blacksquare$	quantification separator	Section 2.1	19
$\langle a, b\rangle$	a sequence of elements a and b	Section 2.1.1	19

Continued on next page

List of Symbols – continued from previous page

Symbol	Meaning	Definition	Page
X^*	set of sequences over X	Section 2.1.1	19
Γ	scaling function on Probabilistic Process Trees	Section 4.5.1	65
σ	a trace	Definition 1	20
τ	the silent activity	Section 2.2	20
Θ	a stochastic language	Definition 2	20
Pr	a probability function	Section 2.1.3	20
$\mathbf{wa}$	gives the WSFA for a PPT	Definition 25	47
$\mathbf{tg}$	the trace language of a model	Definition 3	20
u	a Probabilistic Process Tree	Chapter 4	41
GSPN	Generalized Stochastic Petri Net	Definition 9	22
PLPN	Place-labelled Petri Net	Definition 40	133
PPT	Probabilistic Process Tree	Definition 25	47
SDFA	Stochastic Deterministic Finite Automaton	Definition 10	23
SFA	Stochastic Finite Automaton	Definition 10	23
SLPN	Stochastic Labelled Petri Net	Definition 8	22
WSFA	Weighted Stochastic Finite Automaton	Definition 21	43

Acknowledgements

Thanks to all of the following people who helped me on this journey.

My supervision team, and publication co-authors, Professor Sander Leemans, Professor Moe Wynn, and Professor Arthur ter Hofstede, for their high standards and thoughtful criticism throughout. Both Sander Leemans and Moe Wynn had turns as active and engaged primary supervisors. Particular thanks go to Sander Leemans for his sustained and energetic involvement, even after moving to another continent.

My family, for their love, support, and distractions.

Professor Wil van der Aalst, the godfather of process mining, who helped co-author the publications on quality dimensions for stochastic models, reported on in Chapter 6.

Dr Artem Polyvyanyy, for pointing out that the entropic relevance measures used in Chapter 6 could be straightforwardly extended to any model whose stochastic language was known, and for feedback on the discovery techniques in Chapter 3.

Professor Cameron Campbell, my co-author on work applying process mining to Qing civil service data, reported on in Chapter 7. Also Dr Chen Bijia and the other members of the Lee-Campbell Group who shared insights on the fascinating CGED-Q dataset.

Adam Banham, who contributed to and collaborated on process mining tools for the Python language, including some of those in Appendix A.

Dr Nick Kelly, who understood why I was quoting Deleuze in a thesis on process mining.

All of the smart young bin chickens in the QUT IS Reading Group, and the other students and staff in the School of Information Systems.

This thesis has benefited from the brilliance of all of the people above. Any remaining mistakes are my own.

This research was supported by an Australian Government Research Training Program Scholarship with Fees Offset, and by research assistant funding from the QUT Centre of Data Science. Travel to RWTH Aachen and the 2022 Process Mining Summer School was supported by the PADS and BPM groups at RWTH Aachen, the QUT Short Term Mobility Program, and the Process Science group at QUT. Computational resources used included those provided by the eResearch Office at QUT.

The study conducted in Chapter 7 of this thesis was approved by the QUT Human Research Ethics Committee, project 4930, approval number 2022-4930-11779.

Chapter 1

Introduction

If this is the beginning of the process, everything is also over at this point. All the weaver now has to do is run the program woven in advance.

Sadie Plant
Zeroes and Ones [125]

The most basic question in process mining [6] is simply: What happened?

Process mining is a problem of unsupervised learning of hidden structures. Some process has been going on. Lots of facts are available. We use them to build an image of what happened.

Sometimes, in the world, our materials can show us things that happened. Figure 1.1 is a photograph of some stone steps within Wells Cathedral in Somerset. People have been walking up these handsome steps for more than seven hundred years, making choices about whether to go straight ahead or to continue spiralling up the stairs to the right, subtly adjusting their path accordingly. The behaviour of the visitors is revealed to us in the stone.

Not all materials show us the behaviour of their users so straightforwardly. However, in an era of modern computing, data can be used to create descriptions of behaviour through indirect means. Process mining works with sequential data, like the steps of people ascending a staircase. When you don't have a photograph, process mining can draw you a picture of the paths processes follow. By considering path probability, stochastic process mining [94] gives you more detail: it can show you the wear on the steps.

Though visualisation is important, and we've just used a visual metaphor to introduce it, process mining is not just a data visualisation technique. Process mining visualisations are downstream derivations of *process models*. Models typically have a formal mathematical basis, such as automata, transition systems, or declarative rules. So to be more technically specific, process mining provides ways of computationally interpreting sequential data, and models of that data. The phenomenon being studied is considered a *process*, and the sequential records it produces are called *traces*. A collection of traces is termed a *log*. Typically, traces are considered sequences of *events*, and so logs are usually referred to as *event logs* in process mining work, including in

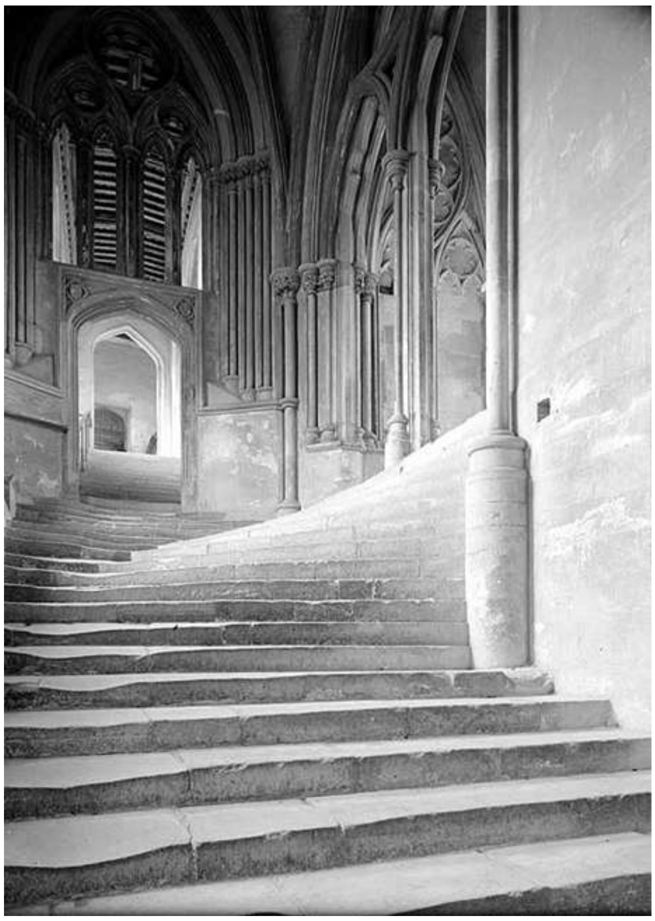


Figure 1.1: Wells Cathedral steps, Somerset, UK, 1948. © Copyright Crown copyright. NMR ref: aa66/00136 [1].

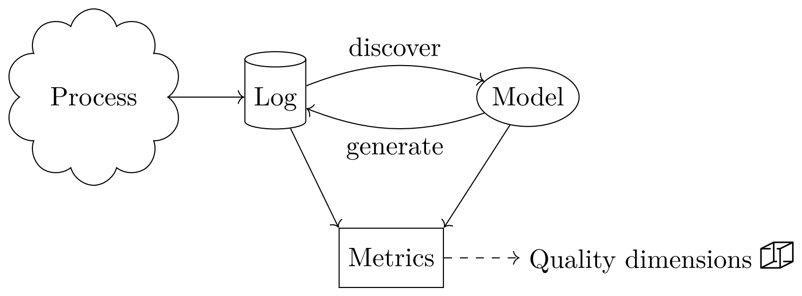


Figure 1.2: High level process mining overview.

Chapters 3-6 of this thesis.

Formal definitions for these terms, as used in this thesis, are in Chapter 2, but the basic ideas are not challenging. When logs are transformed into models, that is *process discovery.* When models are turned into logs, that is *log generation.* When either logs or models are compared, it falls under the topic of *process conformance.* Individual artifacts, or more often, their comparison, can produce summary numbers, and these are *metrics.* There are many process mining metrics [64]. When trying to evaluate model quality, it is useful to categorise metrics according to the underlying regularity or concept they are trying to measure. In process mining, these concepts are termed *quality dimensions.* Figure 1.2 illustrates these different process mining elements. This thesis contributes research in process mining for stochastic models, including discovery, conformance metrics, and quality dimensions.

Stochastic process mining has been identified as a key research challenge [5]. Using stochastic models, we can provide quantitative answers to questions such as *how likely a particular sequence of activities is.* We can do more precise comparisons of models and logs that account for not only the *existence* of paths, but their *probability mass.* Probability is relevant to cost or resource usage calculations, and stochastic process models are used in predictive analytics. Discovering such models automatically provides cheap starting points for such use cases. Such models automatically incorporate special cases represented in large input data sets, even if the models are later refined by a human analyst.

This chapter introduces the thesis. Related research is surveyed in Section 1.1. Research questions are laid out in Section 1.2. Criteria for successful solutions are advanced in Section 1.3, and the research approach in Section 1.4. Section 1.5 describes the real-world data sets used. A summary of contributions, including publications, is in Section 1.6, and Section 1.7 shares an outline of the overall thesis.

1.1 Related Work

We start by informally introducing a key structure in process mining, Petri nets. A detailed literature survey then considers stochastic process discovery, conformance metrics, quality dimensions, and applications. Different data structures change what can be easily calculated or represented, an effect known as representational bias [6, p118], and we review process mining representations along with process mining techniques. Lastly consider work in neighbouring fields.

1.1.1 Petri Nets as Process Models

Before proceeding with the survey of related work, it is worth previewing a modelling structure heavily used in process mining, as well as in every chapter of this thesis: the Petri net [6, p78-83],[20]. Petri nets are a family of nets made up of places (shown as circles or ovals) and transitions (shown as rectangles), where state is indicated by tokens occupying places, and possible changes of state are controlled by tokens proceeding through transitions via their directed edges. They are considered the "oldest and best investigated process modeling language allowing for the modeling of concurrency" [6, p59]. Figure 1.3 shows a Petri net process model. This is a simple insurance claim process with claims being approved or rejected, and activities to close a claim and advise the client who made it. This particular model is a *stochastic model*, specifically, a Stochastic Labelled Petri Net (SLPN) [100], where the numeric weights can be used to derive transition probabilities. This model also contains silent transitions, which are not observable, and are conventionally labelled with τ. Formal specifications of Petri nets and important variants are in Definitions 4,7,8,9, and 40.

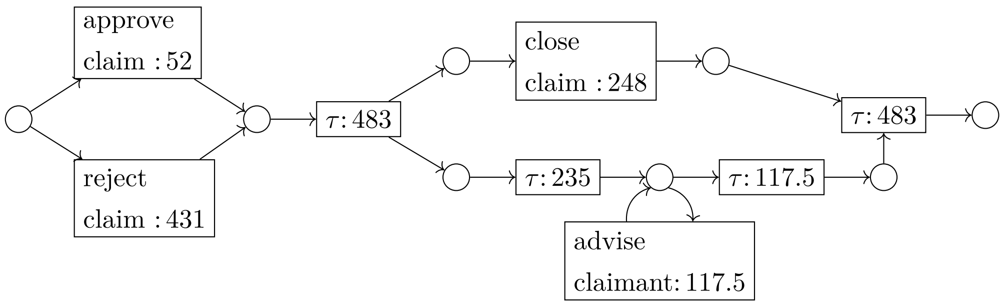


Figure 1.3: A Petri net describing a basic insurance claim process, specifically, a Stochastic Labelled Petri Net (SLPN).

If we take the numeric weights out of Figure 1.3, the model describes what paths are possible, but not their probabilities. This is a common form of *control-flow model* in process mining. Going back to the cathedral staircase, a control-flow model can tell you which steps have been used, but not which are most worn. (*Desire lines* - such as the tracks worn in grass by pedestrians - are a well-established analogy in process mining [6, p43].)

As well as the stochastic perspective, other types of model, with other perspectives, are possible too, such as a data perspective where additional data-aware guards are part of the model, or resource perspectives which track the humans, teams, or machines involved in particular parts of a process. These can also combine interestingly with the stochastic perspective, for example, in understanding the probability that a certain teammate gets a particular task. The focus of this thesis, though, is on process mining with stochastic models.

1.1.2 Discovery

Discovery is fundamental to process mining as it provides the models which all other activities rely on.

Stochastic Finite Automaton Discovery Key formalisms in probabilistic automata are Stochastic Finite Automatons (SFAs) [152] and Stochastic Deterministic Finite Automata (SDFAs) [152]. SDFAs are deterministic in the sense there is no ambiguity in path selection given the next activity in a trace. SFAs are state machines where transitions are governed by a probability function. HMMs can be represented as SFAs [92], as can stochastic forms of Petri nets [66].

Discovery algorithms for SDFAs include the state merging algorithms Alergia [44] and MDL [147]. These work in polynomial time, and Alergia was competitive in the most recent real-world trials [150]. These predate process mining as a field, and pay less attention to problems of conflicting labels or concise human-readable diagrams. The earliest technique we have identified explicitly on the topic of automatic mining for probabilistic workflows builds dependency graphs with probabilities from event log samples [83]. It is from 2000 and draws inspiration from Alergia. These dependency graphs are related to SFAs. Logical deduction and reducing transformations are used in the dependency graph technique, but described at only a high level.

Hidden Markov Model Discovery Another early technique explicitly on the topic of automatic mining for probabilistic workflows is an algorithm for constructing AND-OR graphs from event logs [138], including a mechanic for hidden tasks. Reuse and direct extension of this result is limited by the use of an AND-OR graph representation, the use of small-scale simulated workflow data as input, and advances in process mining art (e.g., concurrency detection) since 2005. This early research does not use the term “event log”. The algorithm implicitly uses a Hidden Markov Model (HMM) [92], a widely studied structure for analysing the probabilities of hidden states.

HMMs are used as the output of separate automated process discovery programs for resource usage [45] and event pathway pruning [12, 13]. The pruning approach [12] uses event log data to prune unlikely event pathways from an initial HMM where all tasks may transition to one another (cousin to the Petri net “flower model” [6, p189 and Fig 6.23]). This may also have applications as a repair technique. Experiments conducted were on healthcare models of seven tasks or less. The HMM structure may be a productive choice for the healthcare domain, where activities may often combine in many different orders. In these domains, it has been noted that process model representations may result in “spaghetti models” [6, p411] with many interconnecting arcs, which can be confusing to visualise and challenging to reason about. By assuming full interconnection up front, HMMs use a kind of spaghetti model by default, which may give them advantages in reasoning in domains with highly interconnected processes, and introduce hard to visualise models that are unnecessarily complex in other settings. In stochastic resource usage discovery [45], HMMs for resource usage per state are built via a frequency matrix of observed direct-follows relations in the event log. This is a useful result but the process model itself is built from a variant of the alpha algorithm, a simple baseline algorithm with known weaknesses when used on real event log data [6, p165]. HMMs have also been used for interactive process exploration [55].

Process Discovery With Stochastics and Control-Flow Within the last decade some techniques explicitly advertise themselves as solutions for stochastic process discovery.

A key contribution to stochastic process discovery, used as a baseline in this thesis, is a technique for constructing Generally Distributed Transition Stochastic Petri Nets (GDT_SPNs) from event logs [130, 129]. GDT_SPNs are a generalisation of Petri nets which allow arbitrary statistical distributions to be associated with the execution time of each transition. The discovery algorithm is a multi-stage process. First a control-flow discovery algorithm is run on the event log. Then alignments [6, p256] are calculated to repair the control-flow model. Finally a statistical distribution estimation procedure is performed over each transition. A public implementation is available. The implementation of this plugin in ProM 6.9 and later uses the Inductive Miner [96, 97] internally as an initial control flow discovery step, which has been updated from the gradient-descent procedure described in [130]. GDT_SPN discovery does not explicitly consider duplicate transition labels or silent transitions, and the implementation does not always produce a model on public reference logs [95].

This reuse of a control-flow model as an input to a stochastic estimation step is shared by a technique discovery of Stochastic Labelled Petri Nets with dynamic weights [99]. The impact of data, in the form of state variables, is the concern of a different estimation technique which produces Data-aware Stochastic Petri Nets [112]. These are Petri nets enriched with data variables and predicate guards, as well as weights on transitions. This structure allows more fine-grained reasoning around key variables, such as the amount of a requested loan.

Stochastic forms of Petri nets are a widely used model representation in this literature survey. They are interoperable with a number of conformance techniques and other computational tools that accept interchange formats such as the Petri Net Markup Language (PNML) [2]. Other models have also been employed. As a means of testing a conformance metric for stochastic process models [126], a discovery technique repurposes the Direct Follows Graph Miner [103], and annotates the result with direct follows frequencies to obtain a stochastic Direct Follows Model. One such technique uses Bayesian networks with non-classical probability [117]. This is the only use of non-classical probability in the survey, though the technique is constrained to exclude loops and concurrency. Bayesian inference by itself yields probability estimates for particular events, but no visualizable model, leading commentators to conclude it cannot be directly applied to process mining [29]. Probabilities obtained with Gibbs sampling [73] have been successfully combined with an input control flow model for stochastic process discovery [86]. The RegPFA framework [29] does prediction by parameter estimation. RegPFA uses an internal model for prediction based on Baum-Welch [19]. It outputs a noise-filtered Petri net model, which emphasizes understandability against precision, and elides stochastic information.

Construction of models for complex simulation of business processes can include stochastic process discovery. Simod [38] is a tool for discovering Business Process Modelling Language (BPMN) models for business process simulation. The model produced includes stochastic and time annotations, calculated by replay and alignment of the event log, and therefore does a form of stochastic process discovery by estimation. The integration of resource, time and activity models is a key goal and the simulation has been refined using deep learning techniques [37].

Queue discovery in stochastic process mining has been investigated using two separate formalisms, Process Trees [134] and Queue-Enabling Colored Stochastic Petri Nets (QCSPNs) [135]. The Process Tree approach is informed by statistics theory and uses both Bayesian and Markov-Chain Monte-Carlo fitting. QCSPNs are a novel construct allowing projection into Queuing Networks and exploitation of analysis results for those structures.

Declarative Models Process mining techniques exist that are not centred on control-flow models. Declarative workflows [4],[6, p266] are expressed in terms of logical constraints on a process: a certain combination of things is not allowed, but everything else is. Control-flow models and their stochastic extensions, as discussed above, express everything that is allowed, with everything else being disallowed. Unlike Petri nets and other graph-based models, declarative models do not have a widely used and accepted visualisations.

Techniques for automatic process discovery of probabilistic declarative models have also been proposed [21, 22, 108]. This includes extending the constraint-based language, DECLARE to a probabilistic variant PROBDECLARE [109].

1.1.3 Conformance

It is one thing to obtain a process model; it is another to know if it's any good. Measuring the quality of models against either logs, or other models, is the concern of process conformance. Carmona defines conformance checking as "analysis of the relation between the intended behaviour of a process as described in a process model and event logs" [43, p3], and van der Aalst adds "techniques can also be used for measuring the performance of process discovery algorithms and to repair models" [6, p243]. This section surveys the problem of trace probability, proposed quality metrics and other comparison techniques for stochastic process models.

Trace Probability "If these five things happen after one another, it costs us $200,000: what are the chances?" Calculating the probability of a particular trace through a process model is a practical problem. It also a non-trivial algorithmic problem, named TRACE-PROB [98] (and identified earlier [102]). Though solutions now exist, efficient approaches for broad classes of models are still an active research challenge. Two recent approaches use linear programming [98] and an expectation-minimisation (EM) algorithm on Probabilistic Context Free Grammar trees [157]. As well as having a direct use on models of a domain, trace probability is an input to a number of conformance metrics.

Conformance Metrics The Earth-Movers' Distance measure [95] combines the well-known concepts of Levenshtein string edit cost - in this case in comparing traces - with the Earth-Movers' distance over the possible traces of model and log. As the set of possible model traces can be infinite, both a universal and truncated measure are defined. The truncated measure uses a specific fraction of the probability mass, for tractability.

Entropy can also be used for model quality measurement. In projection-based precision and recall [101], Stochastic Deterministic Finite Automata (SDFAs) are constructed for both log and model. New SDFAs can be computed from a projection of the log over the model, and vice versa, and entropy ratios then provide measures for precision and recall/fitness. These measures are restricted to process models that can be translated to SDFAs. In entropic relevance [14], the process model is considered as a way of encoding the log. The entropy of the resulting encoding is then calculated using trace probability, accounting for the encoding cost according to a background cost model. Three background cost models are provided, yielding three relevance metrics. A trace probability calculation for SDFAs is used, which again restricts target models to those equivalent to SDFAs. Entropy has also been used to formulate behavioural simplicity measures [89] for control flow models.

The Alpha Precision measure [59] uses the stochastic language of the model and the event log, and is based on statistical inferences about the underlying system that generated the log. Model trace probability is aggregated for those traces where the probability of their occurrence in the underlying system exceeds a parameter called alpha significance. Probability in the (otherwise unknown) underlying system is estimated using attributes of the log, including the traces, and reasoning from a Dirichlet distribution. The alpha significance is a proces specific parameter determined empirically, making it hard to use alpha precision to compare quality across different domains.

Statistical tests and association measures for process model and log comparisons also exist, grounded in the bootstrap method [104]. These were proposed recently (2022) and help define what is a significant difference in behaviour between models, or whether there are significant differences between cohorts which participate in a process.

Other Conformance Techniques Moving onto comparative techniques which do not return a single number, the P^2CM method [113] provides a stochastic-aware way to compare process sub-logs. The calculation of process model alignments in a stochastic-aware fashion [24], used for model and log comparison, has also seen some investigation.

1.1.4 Quality Dimensions

A well-accepted approach for evaluating process models is against four competing quality criteria, or dimensions [35], [6, p188]:

- *Fitness.* The model accommodates traces from the log. Fitness corresponds to the idea of *recall* in machine learning literature, and both terms are used in process mining.
- *Simplicity.* Syntactical elegance and parsimonious expression.
- *Generalization.* Representation of a general underlying process, of which the event log is only a sample.
- *Precision.* The model represents elements that are part of the underlying process.

Quantitative measures exist for each of these dimensions [6, p269-272].

Fitness and Precision Take the fitness quality dimension as an example. Let L be a an event log recorded from a particular process, and M a model of that process, as in the high level process mining overview in Figure 1.2. We use some collection, such as a set, to describe the possible traces in log or model. (More exact definitions for logs and trace languages are forthcoming in Definitions 11 and 3.) We measure the size of some trace collection X with $|X|$. A standard definition of fitness [43, p46] is then the size of the intersection of log and model, divided by the size of the log:

$$\textit{fitness} = \frac{|L \cap M|}{|L|}$$

This definition of fitness measures the proportion of log traces covered by the model. Various techniques exist for performing this calculation on non-stochastic models [43, p166-172]. In the case of a stochastic model, the fitness formula no longer straightforwardly applies. A probabilistic transition may not be well-described using simple set intersection. It is hard to represent frequently occurring traces in the log which are described as highly unlikely by the model. These point to open research questions.

The standard definition of precision [43, p50] parallels fitness, and is the size of the intersection of log and model, divided by the size of the model:

$$\textit{precision} = \frac{|L \cap M|}{|M|}$$

This measures the proportion of model traces that correspond to entries in the log. Parallel problems and open research questions apply to precision in a stochastic process setting (beyond better known challenges such as infinite control-flow languages [144]).

These research challenges may be solved by attempting to adapt existing dimensions to a stochastic setting, as has been proposed for fitness and precision [100, 101]. It may indicate that other concepts are needed to describe quality on these models.

In the course of defining two entropy-based measures, eight desirable properties for stochastic precision and recall (fitness) have been proposed [100]. These include determinism, representation-independence and the formal relationship between precision and recall measures. By contrast, the earth movers' measures [95] are informative about both fitness and precision, but fit neatly in neither category.

Simplicity and Generalisation The simplicity and generalisation dimensions also provide definitional challenges in finding stochastic process model analogues. The motivations for using simplicity in process mining are both philosophical and practical. The principle of Ockham's Razor [139, 17] is the philosophical guide [6, p118], and from an empirical point of view, research has shown simplicity is the best proxy for comprehension of models by people [114]. Philosophers and scientists distinguish multiple types of simplicity, including syntactical simplicity (*elegance*) versus ontological simplicity (*parsimony*) and quantitative simplicity (number of things) versus qualitative simplicity (number of types) [17]. One book on the topic is even titled "Ockham's Razors" [139] (plural) and discusses both a *razor of silence* and a *razor of denial*, while also considering alternative, probability-based criteria such as the *Bayesian Information Criterion* (BIC) [70]. Process model simplicity metrics, based on comprehensibility [114], usually count numbers of structural elements, or quantitative simplicity. There is no standard approach of how this applies to stochastic weights or probabilities.

The generalization quality dimension is glossed as “not overfitting” [6, p189] and has been backed by empirical evaluation in various ways [132, 35]. No generalization measure that specifically addresses stochastic process models in process mining was identified in our survey of the literature.

The generalization quality dimension has been represented by an empirically evaluated measure of *behavioural appropriateness* for Petri nets [132]. This behavioural measure was contrasted with a *structural appropriateness* measure and measures for fitness and precision. An alternative notation-independent metric [35] is based on alignments [3],[6, p256]. Other approaches have included negative events [31] and anti-alignments [62].

In neighbouring fields, the concept of generalization has important differences to its use in process mining, which may affect its application to stochastic process models. In a review of statistical measures and machine learning [153], the idea of generalization is introduced with a complex curve that precisely fits data points vs a straight line: “a physicist measuring these data points would argue that it cannot be by chance that the measurements lie almost on a straight line and would much prefer to attribute the residuals to measurement error than to an erroneous model”. These generalization claims for simplicity are common in philosophy of science [79, 17]. That is, simplicity and generalization are aligned. This is in contrast to the process mining theory, where simplicity and generalization dimensions are described as in tension, or even diametrically opposed [6, p189 and Figure 6.22].

1.1.5 Applications and Neighbouring Fields

Significant work has been done on analyzing the performance characteristics of processes described by Stochastic Petri Nets, including in the context of managing business processes and workflows. Some work comes close to stochastic process mining discovery, because the motivation for using a stochastic model can often be to reason about the duration of executing some process, and some statistical distribution of those durations. Techniques requiring a pre-existing process model are treated here as primarily performance or prediction papers, rather than process mining. One technique introduces workflow constraints on Stochastic Petri Nets and applied a block-reduction technique for performance estimates of stochastic business processes [107]; the analysis has since been extended [148]. Other research close to discovery simplifies the performance model in a Generalised Stochastic Petri Net by using aggregation and elimination rules (or structural foldings) [136]. These simplification rules manage problems of over-fitting, and include quantified constraints on the expected error introduced by a particular rule. Performance remains an active area of research, with other prediction algorithms and techniques intended for use with process mining being suggested and used in industrial case studies [131, 25].

Stochastic process mining is focused on stochastic *models*, but another thread of research concerns *uncertain event logs* [123, 120, 122, 52, 28, 121]. In these logs, some attributes are not fully known, and their values may be non-deterministic. A number of approaches characterise the imperfectly known attributes using probability distributions, describing them as *stochastic attributes*. This is motivated by the large number of Internet of Things (IOT) devices now deployed in many different settings, often with unreliable sensors whose output is better characterised stochastically than with singular values [52]. Using these logs as inputs, progress has been made on discovery of Direct Follow Graphs models [123], process conformance [28], trace probability [121], and efficient log representation [122]. Work on uncertain process logs and on stochastic process mining has seen significant progress over the last four years, and there should be good opportunities to synthesise these approaches in future research, for example by discovering stochastic process models from uncertain logs.

Operations Research (OR), like process modelling and mining, is concerned with formal descriptions of processes within and between organizations. Supply chains and manufacturing resource usage are of particular interest. Petri nets and stochastic Petri nets are also well-established formal tools in Operations Research [160, p1, p157]. Various extensions to this basic formalism exist in OR, including introducing batch token consumption [48] and fuzzy logic [93]. The common methodology is modelling by a human first, followed by analytic or computational simulation calculations to optimize the described process. Constructing the model computationally, as in process mining, is not a clear concern of the literature surveyed. On particular display in Operations Research, but also common across many domains, is the use of Petri nets to describe non-terminating processes. Such processes have no terminal state, and often no initial state either, merely a point of initial observation. A supply chain is a ready example. A supply chain continually feeds material into the manufacturing process of a company, and its termination would usually represent only bankruptcy. Process mining and modelling is oriented around a workflow. A workflow deals with one case, and has designated input and output places. A sound workflow net always terminates and leaves no incomplete work in the form of unconsumed tokens [6, p65]. The two views of non-terminating processes versus workflows are similar to the distinction between a long-running web service and the execution of a single request. Where OR may need to solve for the steady state probability at infinity [20], process mining needs the probability of traces which always terminate, as in Section 1.1.3 above. Though these may be complementary views, they also demand different analysis techniques.

Sequence analysis refers to a broad collection of approaches in the social sciences to understand social phenomena that pass through identifiable stages [54], and examples span linguistics, economics, and sociology. Some of this work comes close to process mining concerns, such as a stochastic flow chart model of decision making in the US federal government budget process [119], or models of the career paths of musicians [10].

Studies have noted the production of handcrafted stochastic workflow models in insurance settings [127], as well as demand for better automation for constructing and analysing them [100], that is, for stochastic process mining. Stochastic declarative process mining has been applied to event data on student career path outcomes [21]. In the healthcare domain, a large Australian hospital used the P^2CM method for comparing stochastic processes [113] built from their internal workflow data.

1.1.6 Research Gaps

From this survey of the literature, we can see that there are a number of opportunities in discovery and conformance where new research on process mining with labelled stochastic net models can contribute new understandings, and solutions connected to real-world problems.

There is limited choice in stochastic process model discovery techniques for stochastic control-flow models, so it is difficult to take advantage of the contemporary abundance of process data by automatically constructing models, and the techniques have a number of limitations. Some are prone to complicated models, that can be difficult to reason about [44, 147, 138, 12, 13]. They can have limitations in dealing with conflicting labels [44, 147], or handling loops and concurrency [117]. Some are applicable mostly to queuing problems [135, 134], or reasoning about data constraints [99], or focused on simulation or temporal use cases which require extra inputs and may reduce their application to a purely stochastic process model setting [130, 38].

Furthermore, though the models and techniques are not incommensurable, they do often use different model representations, and they have often not been compared on common quantitative benchmarks. With the articulation of process mining conformance metrics [95, 14, 59, 101], and algorithms to calculate them, such benchmarking and systematic comparison becomes possible. The metrics themselves still have limitations that provide additional challenges. They may require limits on covered probability mass to terminate [95]. Some are restricted only to SDFAs [14, 101], rather than more general stochastic nets, and others do not allow comparison across different domains [59]. Most commonly, they have a dependency on trace probability or a stochastic language [95, 14, 59, 101]. This makes algorithms for calculating trace probability [98, 157] a constraint on stochastic process conformance, and hence process mining more generally. The progress on conformance metrics also points to limits on what, conceptually, they are measuring. This is the problem of stochastic process quality dimensions, where there are both proposals for the translation of control-flow quality concepts to stochastic processes [100, 101], and difficulties in making such a translation.

This survey then highlights research gaps in stochastic process mining, particularly:

- automatic discovery of stochastic process models;
- restrictions on models supported by quality metrics;
- calculating trace probability for conformance on such models;
- common formalisms that allow analysis, reuse, and comparison across data and models; and
- quality dimensions for such models.

1.2 Research Questions

This project is organised around the overall research question:

How can processes in organisations be understood using stochastic models mined from sequential data?

Two lines of enquiry are identified for this research, *stochastic process discovery* (RQ1) and *stochastic process conformance* (RQ2). These have been broken down into five sub-questions.

Firstly, fundamental to process mining is the discovery of process models from event log data. Existing approaches to stochastic process mining have limitations worth surpassing, as explored in Section 1.1.2.

RQ1 How may stochastic process models be discovered automatically?

Two different approaches to stochastic process discovery involve using well-established control-flow techniques, with their attendant representational biases, versus using new algorithms and formalisms that work directly with the input event log. Both are explored in this project.

RQ1.1 How may control-flow discovery techniques be leveraged for stochastic process model discovery?

RQ1.2 How may stochastic models be discovered directly from event logs?

For these two research sub-questions, the input is an event log, and the output is a labelled stochastic net. New algorithms are stated formally and the resulting frameworks are evaluated experimentally.

Data on processes can describe states rather than events, with concurrent states being a particular challenge. Constructing process models from this data can bring the power of process mining to new domains, particularly sequential data from historical settings.

RQ1.3 How may we discover models of processes from records of concurrent states?

This challenge arises in the context of historical career data from Qing dynasty officials. We develop a new miner which outputs a weighted Petri net to describe career paths, including concurrent roles.

Secondly, stochastic process model conformance checking is a research topic that provides quantitative metrics for model quality, as reviewed in Section 1.1.3. Many existing metrics are limited in the models they accept due to difficulties computing trace probability and stochastic languages. Stochastically sensitive measures do not exist for some process mining quality dimensions, and the dimensions themselves need to be re-evaluated. Improved calculations for, and reporting on, stochastic process model quality helps to address this gap.

RQ2 How may conformance metrics for stochastic process models be conceptualised and calculated?

The power of stochastic models is in describing probability, and the probability of a particular trace being supported by a model is valuable information, both in itself, and as an input to other conformance calculations.

RQ2.1 What is the probability of a given trace through a stochastic process model?

Solutions to this problem on a form of weighted process tree, and on labelled stochastic nets, are investigated formally, and implemented in prototype software.

Challenges in applying process mining quality dimensions to stochastic models and measures, and contrasting interpretations from adjacent fields, show a need to explore and articulate quality criteria specifically for stochastic process mining.

RQ2.2 What quality dimensions apply to stochastic process models and logs?

This question is investigated experimentally, and with a statistical analysis, using a large data set of models and metrics.

1.3 Solution Criteria

In addressing the research questions, this thesis aims to contribute solutions which are suitable, conceptual, empirical, formalised, input general, and tractable.

C1 Suitable. Concepts and solutions should be suitable to the specific problems of process mining in a stochastic setting, including whichever novel concepts and solutions are generated in the course of the research.

C2 Conceptual. Concepts and solutions should be sufficiently distinguished from the specific platforms and domain problems they were designed to address. The research should introduce classes of solutions relevant beyond one particular problem, model, or software implementation.

C3 Empirically Grounded. Process mining was invented to describe the behaviour of organisations. Event and activity logs are data from the real world: specifically, sociological data. The techniques and concepts in this work should be grounded in real-world data, preferably from multiple sources and domains.

C4 Formalised. Important properties of structures and techniques are described using techniques of mathematical specification and proof.

C5 Input General. The process mining techniques and algorithms should tolerate broad formal classes of input models and logs.

C6 Tractable. Algorithms and computational techniques should execute in practical timescales on modern hardware. Wherever possible, this should be because they have been shown to have attractive algorithmic complexity. Depending on the problem, this may be polynomial. However a number of important problems in process mining and computer science are useful but super-polynomial: for example calculating alignments is NP-hard [154, 56].

A key research question for this thesis is on process discovery. It has been argued that process discovery algorithms should achieve four maturity stages [158] (quoting):

DM1 The algorithm is well designed;

DM2 The algorithm is validated on real-life examples;

DM3 The algorithm has an established relationship between the log and model quality;

DM4 The algorithm is effective.

These maturity criteria have influenced our suggested solution criteria. Suitable and tractable process mining solutions are well designed (DM1); real-life logs ensure empirical grounding (DM2); and formal definitions and properties help establish relationships between log and model quality (DM3). Effectiveness (DM4), in the sense of succeeding at multiple challenges with novel data sets in applied settings, connects to input generality, suitability and empirical grounding.

1.4 Approach

This research employs techniques from computer science, information systems, and software engineering. Where appropriate, it also employs iterative design approaches.

Data structures and algorithms designed and developed in this work are specified formally using the foundational tools of set theory and logic. Algorithms and data structures thus specified are then implemented in software. The software artifacts are evaluated experimentally and quantitatively against established benchmarks. In other words, it draws heavily on the research methodology of computer science [53].

As well as the top-down techniques of implementation from formal specification, this thesis employs the bottom-up software engineering techniques of iterative improvement on a prototype artifact [51]. Successful software development is an iterative process where both artifacts and understanding grow through repeated incremental improvement. Iterative design and prototyping is also important when working with expert collaborators from other fields. This "iterative circumscription", informed by experts, artifacts and scholarly literature, is influenced by the principles of Design Science Research [149]. Automated unit tests and the other technical disciplines of software engineering support this approach. Understanding the domain specific quirks of data not previously prepared for process mining also calls for techniques of data analysis and modelling.

Lastly, the entire project is itself both iterative and constructive: artifacts, formalisms, and experimental results obtained from each step in the research inform, and often see reuse in, the next step.

Each individual chapter of this thesis explains the research approach used for that specific work in more detail.

1.5 Data

Process mining depends on data. This research uses real-life or realistic data from a number of different domains. This includes a variety of public event logs, a small realistic event log, and a career tracking data source new to process mining. The use of real-life data particularly supports the solution criteria of empirical grounding (**C3**) and input generality (**C5**), and also helps to demonstrate tractability (**C6**).

The International Business Process Intelligence Challenges[1] are competitions where different process mining discovery and analysis techniques are tried out on novel real-world data. The BPIC logs in Table 1.1 are a selection of these public logs, which are widely used in process mining research. Together with the Road Traffic Fines and Sepsis log, these span organisational domains from issue tracking, through policing, to healthcare. Teleclaims is a realistic example log based on real call centre workflows[2]. Teleclaims is designed to exercise different control-flow model constructs, and is useful for demonstrating behaviour on small but non-trivial inputs. Public event logs used are summarised in Table 1.1.

The China Government Employee Database-Qing (CGED-Q) [40, 47] is a unique digital collection of civil service career records from the Qing dynasty. A project to use process mining to analyse promotion paths in these records is the subject of the case study in Chapter 7. Elite Qing dynasty officials often held multiple concurrent positions, and roles were recorded on a quarterly

[1]BPI Challenges are introduced at `https://www.tf-pm.org/competitions-awards/bpi-challenge` and logs are downloadable from `https://data.4tu.nl/`.

[2]The teleclaims log is available at `http://www.processmining.org/old-version/event-book.html`.

Table 1.1: Reference Public Event logs

Log	Traces	Variants	Activities	Events	Domain
BPIC 2013 closed [141]	1487	183	4	6660	Issue tracking
BPIC 2013 incidents [141]	7554	1511	3	65533	Incident tracking
BPIC 2013 open [141]	819	108	3	2351	Incident tracking
BPIC 2018 control [61]	43808	59	7	161296	EU Agriculture policy
BPIC 2018 dept [61]	29297	349	6	46669	EU Agriculture policy
BPIC 2018 reference [61]	43802	515	6	128554	EU Agriculture policy
Road Traffic Fines [105]	150370	231	11	561470	Italian policing
Sepsis [111]	1054	846	16	15214	Hospital diagnosis
Teleclaims [7]	3512	12	11	46138	Example call centre

Table 1.2: CGED-Q Log Subset Statistics, 1830-1904

Palace Exam Filter	State Entries	Cases	Max Concurrent Roles
Top place	375	36	18
Tier 1 & 2 Top 7	11790	3897	5

basis in public gazettes. When prepared for process mining, we call this type of data a *state snapshot log*, and two such logs constructed from CGED-Q data are listed in Table 1.2. The use of process mining on historical data predating modern computing is, to our knowledge, novel.

1.6 Contributions

This thesis makes several contributions in process discovery and conformance on stochastic models. This section lays out the contributions in the form of a thesis overview, and shares the corresponding scholarly publications.

1.6.1 Research Contributions

Chapters 3 and 4 present novel research contributions related to RQ1 on stochastic process discovery. Chapter 3 addresses RQ1.1 by introducing estimators, novel stochastic process discovery functions which plug into existing control-flow discovery techniques. These are described formally and evaluated by experiments on public event logs. Most of these estimators execute in linear time, except one based on alignments, which depends on an exponential approximation. They accept a broad range of event logs and models, in contrast to preceding techniques and public implementations. Chapter 4 introduces a family of novel algorithms for discovering stochastic process models, the Toothpaste Miner Framework. This works on a novel formalism, Probabilistic Process Trees (PPTs), and addresses RQ1.2. Formal properties are shown, including polynomial time complexity. An evaluation is conducted on public event logs against a reference implementation.

Chapters 5 and 6 describe novel research contributions in the area of stochastic conformance checking (RQ2). Chapter 5 provides a novel solution to the trace probability calculation on Probabilistic Process Trees (PPTs), addressing RQ2.1. The solution is described formally and in a prototype implementation. Chapter 6 explores quality dimensions for stochastic process models, addressing RQ2.2. An empirical approach is taken, making use of discovery techniques from Chapters 3 and 4, as well as innovations in process discovery and conformance metrics from other researchers. Designing these experiments yielded useful secondary results. A miner based on a genetic algorithm contributes a discovery technique (RQ1) suitable for use in laboratory settings, the Stochastic Evolutionary Tree Miner, extending an existing process tree miner to use Probabilistic Process Trees. An new algorithm for generating play-out logs from stochastic models is used to approximate stochastic languages, and hence estimates trace probability (RQ2.1). This computationally cheap approximation for stochastic languages enabled the definition and implementation of a large number of exploratory conformance metrics (RQ2) on stochastic process models.

Chapter 7 works with a novel process mining data set, the CGED-Q database on Qing dynasty civil service careers. This provided the challenging setting of process mining from concurrent state information (RQ1.3). To work with this data, the idea of a process mining *log* was extended to that of a *state snapshot log*, with a corresponding formalism. This stimulated the invention of a novel algorithm for their discovery, the State Snapshot Miner. An implementation and visualisation were also developed in partnership with quantitative historians expert in Qing dynasty society.

1.6.2 Publications

This research has been published, or is in the process of publication, in the following scholarly venues.

1. **Adam Burke**, Sander J. J. Leemans and Moe Thandar Wynn. *Stochastic Process Discovery by Weight Estimation.* In: 5th International Workshop on Process Querying, Manipulation, and Intelligence (PQMI), chairs Artem Polyvyanyy, Claudio Di Ciccio, Sebastian Sardina, Renuka Sindhgatta and Arthur ter Hofstede, at 2nd International Conference on Process Mining (ICPM). ICPM Workshop proceedings ed. by Sander Leemans and Henrik Leopold. Cham: Springer International Publishing, 2021, pp. 260–272. This publication addresses RQ1.1, and forms the basis of Chapter 3.

2. **Adam Burke**, Sander J. J. Leemans and Moe T. Wynn. *Discovering Stochastic Process Models By Reduction and Abstraction.* In: Application and Theory of Petri Nets and Concurrency. Lecture Notes in Computer Science. Springer, 2021, pp. 312–336. This publication addresses RQ1.2, and forms the basis of Chapter 4.

3. **Adam T. Burke**, Sander J. J. Leemans, Moe T. Wynn, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede. *Stochastic Process Model-Log Quality Dimensions: An Experimental Study.* In: 4th International Conference on Process Mining (ICPM). IEEE, 2022, pp. 80–

87. This publication addresses RQ2.2, includes a solution for RQ2.1, and is reported on in Chapter 6.

4. **Adam T. Burke**, Sander J. J. Leemans, Moe T. Wynn, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede. *A Chance For Models To Show Their Quality: Stochastic Process Model-Log Dimensions.* Invited paper for Information Systems. Completed second round review. This extension of the previous ICPM 2022 conference paper addresses RQ2.2, includes a solution for RQ2.1, and forms the basis of Chapter 6.

5. **Adam T. Burke**, Sander J. J. Leemans, Moe T. Wynn, and Cameron D. Campbell. *State Snapshot Process Discovery on Career Paths of Qing Dynasty Civil Servants.* Presented at and forthcoming in proceedings for: 2023 5th International Conference on Process Mining (ICPM). This publication addresses RQ1.3, and forms the basis of Chapter 7.

1.7 Outline

This thesis proceeds with formal preliminaries, a chapter devoted to each research question, a closing case study on novel data, and a conclusion. Chapter 2 lays out formal mathematical definitions. Chapter 3 introduces process discovery through weight estimators. Chapter 4 introduces Probabilistic Process Trees (PPTs) and a framework for discovering stochastic process models, the Toothpaste Miner Framework. Chapter 5 provides a solution to the trace probability calculation on Probabilistic Process Trees. Chapter 6 explores quality dimensions for stochastic process models in an empirical study. Chapter 7 works with the CGED-Q database on Qing dynasty civil service careers, considering logs of concurrent state information, and the State Snapshot Miner. Chapter 8 concludes.

All of the novel techniques in this thesis come with public software implementations. These, and related utilities developed in the course of the research, are detailed in Appendix A.

Chapter 2

Preliminaries

In this chapter, we provide formal definitions for concepts used throughout the thesis. We start with foundational concepts such as sequences and probability in Section 2.1. Computer science concepts such as languages, Petri nets and stochastic automata are introduced in Sections 2.2 and 2.3. Finally we provide definitions for logs and from the process mining literature in Sections 2.5.

2.1 Foundations

Throughout this thesis, quantifier variables are separated from their predicates with $\bullet$, e.g., $\forall x \in \mathbb{N} \bullet x \geqslant 1$. Function composition is denoted $f \circ g$ such that $\forall x \bullet (f \circ g)(x) = f(g(x))$. The powerset of a set C is written $\mathbb{P}(C)$.

2.1.1 Sequences

A finite sequence over set X of length n is a mapping $\sigma \in \{1..n\} \rightarrow X$ and denoted by $\sigma = \langle a_1, a_2, ..., a_n \rangle$ where $\forall i \bullet a_i = \sigma(i)$. Concatenation operator $+$ appends one sequence to another such that $\langle a_1, ..., a_n \rangle + \langle b_1, ..., b_m \rangle = \langle a_1, ...a_n, b_1, ..., b_m \rangle$. The tail function is then $tail(\langle a \rangle + \sigma) = \sigma$. $s[i]$ is the ith member of sequence s. We also use slice ($[...\ddagger...]$) and length $|s|$ operations. $s[i \ddagger j]$ is a slice from indexes i to j inclusive, e.g., $\langle a, b, c, d \rangle[2 \ddagger 3] = \langle b, c \rangle$.

The set of all sequences over the set X is X^*. The indexes and values of a sequence are given by dom and ran respectively, similar to function domain and range [140].

Function sqct returns the number of times a subsequence is present in a sequence:

$$\mathrm{sqct}(\varsigma, \sigma) = \begin{cases} 0 & \text{if } \sigma = \langle\rangle \\ 1 + \mathrm{sqct}(\varsigma, tail(\sigma)) & \text{if } \sigma = \varsigma + x \\ \mathrm{sqct}(\varsigma, tail(\sigma)) & \text{if } \sigma \neq \varsigma + x \end{cases}$$

2.1.2 Multisets

The set of multisets (bags) over type C is $\mathcal{B}(C)$. Multisets are shown within [] with superscript frequencies, e.g. $[a^3, b^2]$. The count of item $x \in C$ in bag $B \in \mathcal{B}(C)$ is $B[x]$. Multiset union and intersection are $\uplus$ and $\Cap$ respectively. Multiset subset $C \leqslant D$ overloads $\leqslant$, and requires that all members $c \in C$ must be present in D and have lower counts, $C[c] \leqslant D[c]$. The $\cdot$ operator scales all occurrence values in a multiset by a numeric factor.

Extending that concept, real-valued multisets are shown as $\mathcal{B}^+(C)$. Real-valued multisets are always positive-valued in this work, with values $\in \mathbb{R}^+$. In real-valued multisets, the count of a member is in $\mathbb{R}^+$. As an example, consider real-valued multiset $X = [\langle a\rangle^{3.4}, \langle b, c\rangle^{2.0}]$. Then $X[\langle a\rangle] = 3.4$.

2.1.3 Probability

Throughout, if not associated with a specific function, calculations of probability are shown with Pr, such as $Pr(\text{walk}) = 0.3$.

2.2 Activities and Languages

An activity is an identifiable thing that happens in a process.

Definition 1 (Activities and Traces)**.** *Let A be a set of activities in a process, and A^* the possible finite sequences of those activities. A* trace *$\sigma \in A^*$ is a sequence of activities.*

In process mining data, it is useful to distinguish between an activity, and what it is named in data. This allows reasoning about problems like two distinct activities being named the same thing, or activities without names.

τ is a silent label where $\tau \notin A$. Depending on interpretation, τ may label a purely structural element of a model, or a real event that was unobservable, and so not recorded. The set of all activities is $\mathcal{A}$.

In process mining, distinct traces are identified by means of *case identifiers*, which are constant for a particular trace. The meaning of a case depends on the process. We will often abstract away from this detail. More often, we will work with a stochastic language of possible traces.

Definition 2 (Stochastic Language)**.** *A* stochastic language $\Theta \subseteq \mathcal{B}^+(A^*)$ *for traces over activities A is a real-valued multiset holding probability values for each trace, and summing to 1.*

$$\forall \sigma \in \operatorname{dom} \Theta \bullet \Theta[\sigma] \in [0,1] \wedge \sum_{\sigma \in \operatorname{dom} \Theta} \Theta[\sigma] = 1$$

A *trace language* is the set of valid activity sequences for a representation of a process.

Definition 3 (Trace Language)**.** *Given stochastic language Θ, the corresponding* trace language *TL of allowed traces is $TL = \{\sigma \in \operatorname{dom}(\Theta) \mid \Theta(\sigma) > 0\}$.*

Control-flow models describe trace languages, where stochastic models describe both trace and stochastic languages. We refer to the trace language of model m as $\mathbf{tg}(m)$.

A *path* records the sequence of transitions followed through some model. A path is a sequence of activities and silent activities, so for activities $A \subseteq \mathcal{A}$, path $p \in (A \cup \{\tau\})^*$. When dealing with stochastic models, paths may sometimes be shown with associated numeric weights for clarity.

Function *trace* gives the trace for a path by removing silent activities. For example, $trace(\langle a, \tau, b \rangle) = \langle a, b \rangle$.

2.3 Petri Nets and Their Extensions

Many models in this thesis are Petri nets, which we described informally in Section 1.1.1. The term Petri net can refer either to a specific formalism, or to a family of related transition structures. We refer to the foundational control-flow structure as a place-transition net, following existing usage [20].

Definition 4 (Place-transition nets). *A place-transition net is a tuple $PN = (P, T, F, M_0)$, where P is a finite set of places, T is a finite set of transitions, and $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation. $P \cap T = \varnothing$. A marking M is a multiset of places $M \in \mathcal{B}(P)$ that indicate a state of the net, with $M_0 \in \mathcal{B}(P)$ the initial marking.*

The set of all place-transition nets is $\mathcal{PN}$. The set of all transitions in $\mathcal{PN}$ is $\mathcal{T}$ and the set of all places $\mathcal{P}$. For a node $n \in P \cup T$ in net (P, T, F, M_0), we define predecessor and successor operations $\bullet n = \{y \mid (y, x) \in F\}$ and $n\bullet = \{y \mid (x, y) \in F\}$.

A transition t is *enabled* if all the places p where $(p, t) \in F$ contain a token.

Definition 5 (Place-transition enablement). *Function eb determines which transitions are enabled in a given marking.*

$$eb \colon \mathcal{PN} \times \mathbb{P}(\mathcal{P}) \to \mathbb{P}(\mathcal{T})$$
$$eb((P, T, F, M_0), M) = \{t \in T \mid \forall p \in P \bullet \bullet t \leqslant M\}$$

A transition *fires* by changing the marking of the net to consume incoming tokens and produce tokens for its outgoing transitions.

Definition 6 (Place-transition firing rule). *Given net $(P, T, F, M_0) \in \mathcal{PN}$, a firing relation $fire \subseteq \mathcal{B}(P) \times T \times \mathcal{B}(P)$ describes which transitions can fire for a given marking, and the new marking after they fire. It is the smallest relation satisfying the condition:*

$$\forall m, t, m' \bullet (m, t, m') \in fire \iff t \in eb(m) \wedge m' = (m \setminus \bullet t) \uplus t\bullet$$

In process mining, it is often useful to distinguish activities from the log from transitions in a discovered model, for instance to tolerate duplicate or silent labels.

Definition 7 (Labelled place-transition nets). *A labelled place-transition net is a tuple $PN = (P, T, F, M_0, \lambda)$, where (P, T, F, M_0) is a place-transition net. Function $\lambda \colon T \to A \cup \{\tau\}$ associates each transition with a label.*

This thesis is on stochastic process mining, and many of the models used are Petri nets with both stochastics and labels. In an Stochastic Labelled Petri Net, transitions are given weights, which are used to calculate transition probabilities.

Definition 8 (Stochastic Labelled Petri Nets (SLPNs))**.** *An* SLPN *[100] is a tuple* $(P, T, F, M_0, W, \lambda)$ *such that* (P, T, F, M_0, λ) *is a labelled place-transition net. A weight function* $W \colon T \to \mathbb{R}^+$ *assigns each transition a weight. Labelling function* $\lambda \colon T \to A \cup \{\tau\}$ *then provides a mapping from transitions to a symbol library of activities* A*.* τ *is a silent label where* $\tau \notin A$*.*

When transitions $T_e \subseteq T$ *are enabled in a particular marking, a transition* $t \in T_e$ *fires according to the probability given by* $\frac{W(t)}{\sum_{t' \in T_e} W(t')}$*. The sequences of labels generated by a series of transitions through the model forms a trace, and the collection of such traces and their probabilities is the SLPN's stochastic language. We assume traces to end in deadlock, where no transitions are enabled.*

The set of all SLPNs is $\mathcal{SN}$.

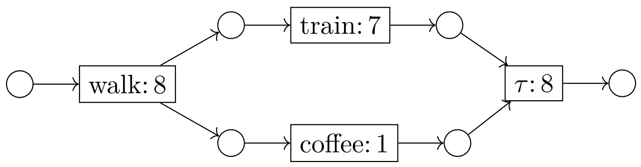


Figure 2.1: Example SLPN. An SLPN with weights removed is a labelled place-transition net.

Figure 2.1 shows an example SLPN for a simple travel scenario. A person taking the train to work first walks to the train station. Sometimes they get a coffee before they embark, at the cafe near the train station, and sometimes after they arrive on the train, from the cafe near their office. They usually take the train before buying their coffee, so that activity has a higher relative weight. Weights are shown on transitions after labels. Whether the coffee or train activity is first is modelled as the outcome of a weighted race, using a Petri net concurrency construct where two tokens will be produced from the *walk* transition. The commuter has a $\frac{7}{8}$ chance of taking the train before buying their coffee, and only a $\frac{1}{8}$ chance of buying a coffee first.

Definition 9 (Generalized Stochastic Petri Net (GSPN))**.** *A Generalized Stochastic Petri Net (GSPN) [20] is a tuple* $(P, T, F, M_0, W, T_i, T_t)$ *such that* $T_i \subseteq T$*,* $T_t \subseteq T$ *and* $T_i \cap T_t = \varnothing$*. Weight function* $W \colon T \to \mathbb{R}^+$ *assigns each transition a weight.* T_i *is a set of immediate transitions. If multiple transitions* $T_e \subseteq T_i$ *are enabled in a particular marking, the probability of a transition* $t \in T_i$ *firing is given by* $\frac{W(t)}{\sum_{t' \in T_e} W(t')}$*.* T_t *is a set of timed transitions. Immediate transitions take priority over timed transitions. A timed transition, if enabled, fires according to an exponentially distributed wait time. Given a set of enabled timed transitions* $T_e \subseteq T_t$*, a particular transition* t *fires first with probability* $\frac{W(t)}{\sum_{t' \in T_e} W(t')}$ *[11].*

The set of all GSPNs is $\mathcal{GN}$. Note that an SLPN is a GSPN with a labelling function and only immediate transitions. A GSPN with only timed transitions is called an Stochastic Petri Net, or SPN [20]. Both SPNs and GSPNs are well-studied and have industrial applications. Some process mining work has worked with GSPN extensions [130, 131]. In this thesis, the main focus is on labelled models with abstracted timing sequences such as SLPNs. The exact duration of processes are not a focus, and so the timed transitions of SPNs and GSPNs introduce unnecessary complexity. The mix of timed and untimed transitions in GSPNs also have subtle semantic implications [67, 66]. Standard treatments of these models also do not consider the naming problems important in a process mining setting. Labels on transitions may possibly be duplicated, or absent, indicated by labelling with the silent activity. Most of the solutions for SPNs and GSPNs also reason about the dynamics of the net under infinite behaviour, rather than the terminating traces of process workflows. The SLPN is accordingly the main net formalism used in this research, though the results may also shed light on problems involving GSPNs.

2.4 Automata

Stochastic Finite Automata [152], also called Probabilistic Finite Automata [152, 67], are a well-studied formalism consisting of states, and transitions between them, governed by a probability function with static odds. Our definition is drawn from literature on common semantics for stochastic models [67].

Definition 10 (Stochastic Finite Automata)**.** *A* Stochastic Finite Automata *(SFA) [151, 67] is a four-tuple* (S, Act, GS, s_0)*, where*

- S *is a non-empty, finite set of states,*
- $Act \subseteq \mathcal{A} \cup \{\tau\}$
- $GS \subset S \times Act \times S \to [0, 1]$ *is a transition function, from one state to another, generating a symbol, at a given probability. GS fully connects the states in* S*.*
- $s_0 \in S$ *is the initial state*

States from which no transition exists are called *terminal.* Probabilities for transitions exiting a non-terminal state sum to 1.

When a transition fires, it generates the corresponding symbol. The sequence of such symbols generated from the initial state is a trace in the language of the SFA. Time is treated as discrete, distinguishing the sequence of firing events, and otherwise not modeled. Probabilistic Automata are Markov Automata [67] where there are no timed transitions, and possible probability distributions are restricted as above.

Figure 2.2 shows an example SFA for a simple travel choice between walking all the way to a destination, or driving partway before walking the final leg. Probabilities are shown next to actions.

When SFAs are deterministic, they are called Stochastic Deterministic Finite Automata (SDFAs) [152]. Determinism indicates that a transition from a state can be uniquely identified if the symbol is known. Figure 2.2 is an SDFA. A number of results are applicable to SDFAs, but not SFAs in general, including the entropy precision and recall conformance measures [101] in process mining.

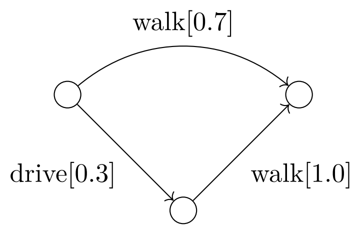


Figure 2.2: Example SFA model for travel choices.

2.5 Process Mining

These definitions draw on the process mining literature [6, 95].

2.5.1 Logs

Records of processes, in the form of traces, are collected into *logs*.

Definition 11 (Event Logs)**.** *Let A be a set of activities in a process. A transitory occurrence of an activity is an* event*. An* event log *for the process is a multiset of traces $\mathcal{B}(A^*)$.*

The number of traces in a log L is denoted with $|L|$, while the number of events is denoted with $||L||$. The set of all event logs is $\mathcal{L}$. The number of cases matching trace σ in log $L \in \mathcal{L}$ is $L[\sigma]$.

It is a well-established convention in process mining texts [6, p32] to make events an occurrence of an activity. Chapter 7 discusses a process mining case study where, in order to model a new type of real-world process data, trace elements may be persistent states, named *roles*, rather than transitory events.

2.5.2 Conformance

Process conformance is the area of process mining concerned with comparison and measurement. This includes measuring single artifacts such as models or logs, but another important concern is comparing one model with another, and comparing a model for a process to a log from that same process.

In this work, *metrics* and *measures* are particular classes of functions for describing some aspect of a model with a single number.

Definition 12 (Metrics and Measures)**.** *A* metric *m is a function comparing models and logs, $m\colon \mathcal{SN} \times \mathcal{L} \to \mathbb{R}$, which returns a real number. A* measure *μ is a metric with range $[0,1]$, or $\mu\colon \mathcal{SN} \times \mathcal{L} \to [0,1]$.*

It is useful, but non-trivial, to determine the probability of traces produced by stochastic process models. This has been formulated as the Trace-Prob problem [98].

Definition 13 (Trace Probability)**.** *Let $A \subseteq \mathcal{A}$ be the activities for a given SLPN. Then function π calculates the probability of the model supporting a particular sequence of activities (trace).*

$$\pi\colon A^* \times \mathcal{SN} \to [0,1]$$

Though we use a definition based on SLPNs, more general variants on stochastic transition systems are also discussed in the literature [98]. Note that having a full stochastic language for a model is equivalent to calculating the trace probability for every possible trace.

When comparing each trace from a log with a model, the cases where traces fit completely, or not at all, are relatively straightforward. More challenging is meaningful description of when traces partially fit a model. Alignments are an established solution for this using a formalised ”skip” of a step in trace or model, called a log move or model move. A cost function makes this choice systematic.

Definition 14 (Alignments)**.** *An alignment [3],[6, p256] represents paired paths between a log and a model. That is, a move is a tuple where* (a, t) *represents a synchronous move on activity* a *in a trace and a transition* t *in the model (with the same label:* $\lambda(t) = a$*),* $(a, \perp)$ *represents a log move, and* $(\perp, t)$ *represents a model move.*

Alignments are a NP-hard problem [154, 56], but with established optimisations and tooling. In this thesis we reuse these formal and tooling components.

2.6 Summary

The following chapters make use of the foundations described above in introducing a number of stochastic process mining solutions.

Chapter 3

Discovery By Weight Estimation

He did not understand - none of these people did - that this is *stochastic* land on the edge of a *stochastic* reservoir.

Neal Stephenson
Termination Shock [142]

Process discovery techniques have become quite sophisticated at describing relationships between activities from sequential event logs, and representing that in process models. There are far fewer techniques for discovering relative probabilities (discussed in Section 1.1). This chapter describes a framework for automatic discovery of stochastic process models, given a control-flow model and an event log.

The framework in Section 3.2 leverages existing art by allowing transformation of models describing only control flow into stochastic process models. Specifically, it takes a Petri net model and an event log as input, and outputs a Stochastic Labelled Petri Net (SLPN) [100]. This extends an existing stochastic process discovery technique [130, 131], in two ways. Firstly, it generalises one estimation algorithm to a general class of *weight estimators*. Secondly, it adds support for silent transitions. Thirdly, it specialises the possible outputs from general probability distributions to SLPNs. The framework does not prescribe whether the estimation calculation is deterministic, uses stochastic simulation, or other techniques, and our introduced estimators include both deterministic and non-deterministic types. Using the framework, six new weight estimators are introduced, and a method for their evaluation.

This approach is a form of stochastic process discovery, as it takes an event log input and produces an SLPN output. In decoupling weight estimation from control flow discovery, the technique also shares some features with process model enhancement for time and probability [6, p290]. Unlike enhancement techniques, estimators can potentially change control flows when producing a stochastic process model. The algorithms and models presented here are evaluated using stochastic process conformance measures. Evaluation, which also includes performance, is against real-life event logs, multiple control flow discovery algorithms, and GTD_SPN discovery [130]. The algorithms have been implemented in the open-source process mining framework

ProM. Using stochastic conformance measures, the resulting models have comparable conformance to existing approaches and are shown to be calculated more efficiently.

In the remainder of this chapter, related work is reviewed in Section 3.1. In Section 3.2, we describe the weight estimation framework, and in Section 3.3 instantiate it by introducing novel estimators. Section 3.4 presents an experimental design using the estimators on real-world event logs, and Section 3.5 shares the results. Section 3.6 discusses estimator design limitations. Section 3.7 concludes the chapter.

3.1 Related Work

In the process mining literature, a number of techniques leverage the success of control-flow discovery algorithms by adding stochastic information to a control-flow model [130, 86, 112]. GDT_SPN discovery [130] is a technique, with publicly available implementation, for discovering Generally Distributed Transition Stochastic Petri Nets (GDT_SPNs). GDT_SPN first discovers a control flow model in the form of a Petri net, then performs a fitness calculation, and attempts to repair the model if fitness is low. An alignment and replay calculation then informs the production of an output GDT_SPN. The distinction between control flow discovery and stochastic perspectives is extended by our proposed framework to many possible weight estimators. The output GDT_SPN is treated as a SLPN in these benchmarks.

Other techniques use Markov Chain Monte Carlo (MCMC) [86] or distribution fitting with a goal of business process simulation [38]. MCMC [86] discovery uses the stochastic model internally, with the output model being non-stochastic. Simod requires more complex input event logs, including information on resources, to construct an output model in BPMN with stochastic annotations.

Our experimental evaluation in Section 4.6.2 makes use of quality metrics from the literature. The Earth-Movers' Distance (EMD) measure [95] represents shared probability mass between models or between logs and models, building on the well-known Levenshtein string edit distance. Measures for entropy precision and recall (fitness) [101] have been defined using Stochastic Deterministic Finite Automata [152].

Entropic relevance [14] and Alpha-precision [59] are measures described in the literature after these experiments were designed and initially conducted. They are used in Chapters 4 and 6.

3.2 A Framework for SLPN Discovery

The framework defines functions which together transform an event log into a SLPN, as shown in Figure 3.1 and defined formally below.

Control-flow discovery produces a labelled place-transition net model from an event log. SLPN discovery produces an SLPN model from an event log. Estimators take a control flow model and a log and return an SLPN model. If the underlying control-flow appears unchanged by estimation in the resulting SLPN, these are called simple weight estimators. The signature reflects the only new output being the weight function for transitions. A SLPN can be constructed by combining

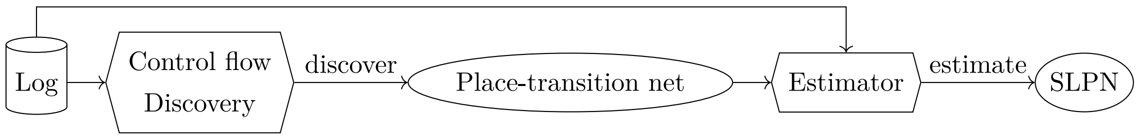


Figure 3.1: Framework for SLPN Discovery with estimators.

the weight function with the original place-transition net.

Definition 15 (Control-Flow and SLPN Discovery)**.** *We introduce terminology for functions with the following properties and signatures*

$$\begin{aligned} cfd &: \mathcal{L} \to \mathcal{PN} && \textit{termed} \text{ control-flow discovery} \\ mine\_slpn &: \mathcal{L} \to \mathcal{SN} && \textit{termed} \text{ SLPN discovery} \\ est &: \mathcal{L} \times \mathcal{PN} \to \mathcal{SN} && \textit{termed} \text{ estimators} \\ se &: \mathcal{L} \times \mathcal{PN} \to (\mathcal{T} \to \mathbb{R}^+) && \textit{termed} \text{ simple weight estimators} \\ &\forall (P_d, T_d, F_d, Md_0) \bullet T_d = \operatorname{dom} se(l, (P_d, T_d, F_d, Md_0)) \end{aligned}$$

As seen in Figure 3.1, our framework considers functions of the form $mine\_slpn = est(cfd(L), L)$, where the log used to discover the control flow is reused to estimate stochastic properties of the resulting SLPN.

3.3 Estimators

Specific estimators may have further restrictions on their inputs, or provide guarantees on their outputs. For example, most of the estimators discussed below do not distinguish transitions with duplicate labels. The Alignment Estimator in Section 3.3.5 is the exception. A challenge common to several estimators is treatment of silent transitions, as those transitions in a discovered model serve a structural role and do not directly represent an activity in the log. Assigning such a transition a weight of zero in a stochastic net is equivalent to deleting the transition, and all subsequent model paths. To avoid this impact, default values are assigned to silent transitions where the calculation would otherwise result in zero weights. In general, these estimators make no distinction between silent transitions and transitions without a corresponding activity in the log. Other trade-offs in estimator design are discussed with experimental results in Section 3.6.

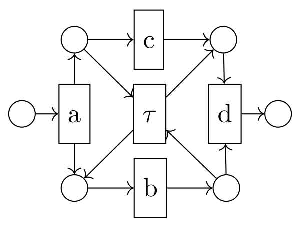
In the remainder of this section, we introduce six simple weight estimators that instantiate this framework. GDT_SPN discovery is an example of a non-simple estimator, as it may change the original control-flow structure based on the result of calculated alignments.

3.3.1 Frequency Estimator

The first estimator, w_{freq}, straightforwardly uses how often each the activity label for transition t appeared in the event log L.

$[\langle a,b,c,d\rangle^5,$
$\langle a,c,b,d\rangle^4,$
$\langle a,b,b,d\rangle^2,$
$\langle a,b,c,b,d\rangle]$

(a) Log EL.



(b) Petri net EPN.

	w_{freq}	w_{lhpair}	w_{rhpair}	$w_{pairscale}$	w_{fork}	w_{align}
a	12	12	12	$1\frac{11}{49}$	12	12
b	15	8	7	$\frac{35}{49}$	8	14
c	10	4	5	$1\frac{1}{49}$	$4\frac{10}{11}$	9
d	12	12	12	$1\frac{11}{49}$	$11\frac{6}{13}$	12
τ	1	1	1	1	$\frac{129}{143}$	2

(c) Six example estimators.

Figure 3.2: Running example of an event log and a Petri net, and the estimators.

Definition 16 (Frequency estimator)**.**

$$w_{freq}(L,(P,T,F,M_0))(t) = \max(1, \Sigma_{\sigma \in L}\,\text{sqct}(\langle\lambda(t)\rangle,\sigma))$$
$$\textit{where } t \in T$$

Silent transitions are assigned the arbitrary weight of 1, equivalent to a single observation in the log. The complexity of this estimator is linear in the number of events in the log. Figure 3.2c shows the results of this estimator on our running example, e.g. $w_{freq}(EL,b) = 15$. The w_{freq} estimator ignores the structure of the log and is included as a deliberately simple baseline technique.

3.3.2 Activity-Pair Frequency Estimators

An Activity-Pair Estimator uses the frequency of pairs of successor activities to better reflect the constraints of more general Petri nets. These are *edge-structured estimators*, in that Petri net edges inform the weighting.

We first introduce some frequency definitions. The functions $freq_I$ and $freq_F$ capture how often the label of a transition appears as the first/last in a trace. The function $freq_P$ takes two transitions, and captures the frequency of their labels appearing as activity pairs in the log. Put another way, it counts when the two activities follow one another directly in the log.

$$freq_I(L,t) = |[\langle\lambda(t),\ldots\rangle \in L]|$$
$$freq_F(L,t) = |[\langle\ldots,\lambda(t)\rangle \in L]|$$
$$freq_P(L,s,t) = \Sigma_{\sigma \in L}\,\text{sqct}(\langle\lambda(s),\lambda(t)\rangle,\sigma)$$

There are both left-handed and right-handed variants of the Activity-Pair estimator, depending on whether weights are informed by successor or predecessor transitions, defined as:

Definition 17 (Left and Right Activity-Pair Estimators)**.**

$$w_{lhpair}(L,(P,T,F,M_0))(t) = \max(1, freq_I(L,t) + freq_F(L,t) + \sum_{s \in \bullet(\bullet t)} freq_P(L,s,t))$$
$$w_{rhpair}(L,(P,T,F,M_0))(t) = \max(1, freq_I(L,t) + freq_F(L,t) + \sum_{s \in (t\bullet)\bullet} freq_P(L,t,s))$$
$$\textit{where } t \in T$$

There are no restrictions on input Petri nets and they can be calculated in time $O(||L|| \cdot |F|)$, that is, the number of events times the number of model edges.

When using activity pair frequency data, two important types of path through the model are neglected for any given trace: paths from the initial place to the first transition, and the paths from the last transition to the final place. Traces of length one are also invisible from this perspective. To account for this, how often an activity appears as the initial or final activity in a trace is also included in the weight estimation. Note that not all activity pairs occurring in the log are used to calculate the resulting transition weights. For instance, where a given Petri net represents two transitions a and b as concurrent and do not share an input place in the model, the frequency of $\langle a, b\rangle$ will not be used. In the running example in Figure 3.2c, $w_{lhpair}(EL, c) = 4$ and $w_{rhpair}(EL, c) = 5$.

3.3.3 Mean-Scaled Activity-Pair Frequency Estimator

The previous estimators depend on the size of the log. Two logs with the same traces in the same ratios will result in two models with two distinct sets of weights, which challenges human analysis. Though comparison and comprehensibility of stochastic process models appears not to have been directly addressed in the literature, it is consistent with research that finds "small variations between models can lead to significant differences in their comprehensibility" [115] and the usability principle of minimizing user memory load [118, p129]. The mean-scaled activity-pair estimator $w_{pairscale}$ mitigates this effect by scaling weights by average transition frequency ($\frac{||L||}{|T|}$) in the log L.

Definition 18 (Mean-Scaled Activity-Pair Frequency Estimator)**.**

$$pairscale(L,T,t) = \frac{freq_I(L,t) + freq_F(L,t) + \sum_{s \in (t\bullet)\bullet} freq_P(L,t,s)}{\frac{||L||}{|T|}}$$

$$w_{pairscale}(L,(P,T,F,M_0))(t) = \begin{cases} pairscale(L,T,t) & \textit{if } pairscale(L,T,t) \neq 0 \\ 1 & \textit{otherwise} \end{cases} \qquad \textit{where } t \in T$$

One effect of defaulting after scaling is that silent or unrepresented transitions are weighted more heavily, that is, the same as an activity of mean-frequency, rather than the equivalent of an activity occurring once in the log. In our running example of Figure 3.2c, $||L|| = 49$, $|T| = 5$ and the numerator of *pairscale* is equal to w_{rhpair} for a, b, c and d. Then, for instance $w_{pairscale}$ of c is $\frac{10}{\frac{49}{5}} = 1\frac{1}{49}$.

3.3.4 Fork Distribution Estimator

The Fork Distribution Estimator w_{fork} uses a two-stage approach: it first assigns weights to each place in a Petri net using activity-pair frequencies. Second, it distributes those weights to transitions according to the activity frequency in the event log.

Definition 19 (Fork Distribution Estimator)**.**

$$\begin{aligned} pw(L,p) &= \begin{cases} |L| & \textit{if } p \in M_0 \\ \Sigma_{s \in \bullet p} \Sigma_{t \in p \bullet} freq_P(L,s,t) & \textit{otherwise} \end{cases} \\ placeWeights(L,p) &= \max(1, pw(L,p)) \\ w_{fork}(L,(P,T,F,M_0))(t) &= \Sigma_{p \in \bullet t}\, placeWeights(L,p) \frac{w_{freq}(t)}{\Sigma_{t' \in p \bullet} w_{freq}(t')} \qquad \textit{where } t \in T \end{aligned}$$

This estimator only applies to Petri nets that have at least one place without incoming edges, such as workflow nets [6, p81]. This is an edge-structured estimator informed by the structure of the input net. The complexity is $O(||L|| \cdot |F|)$. The w_{fork} estimator shares similarities with the Alpha algorithm [6, p167], in that it treats a place as defining a neighbourhood of related activities represented as transitions. In our example in Figure 3.2, let p_1 be the top-right place and p_2 the bottom-right place. Then, $pw(EL,p_1) = freq_P(c,d) + freq_P(\tau,d) = 5$, $pw(EL,p_2) = freq_P(\tau,d) + freq_P(b,d) = 7$, $placeWeights(EL,p_1) = 5$, $placeWeights(EL,p_2) = 7$ and w_{fork} of $d = 5\frac{12}{12} + 7\frac{12}{13} = 11\frac{6}{13}$.

3.3.5 Alignment Estimator

The estimator w_{align} applies alignments [3] to estimate weights. To this end, it counts the number of times a transition t appears either as a model move or as a synchronous move in the alignments. For our purposes, we assume that a function *align* is available for calculating alignments as described in Definition 14. The *align* function takes a Petri net, a set of final markings and an event log, and returns a sequence of move tuples that represent all moves necessary to align every trace in the log.

Definition 20 (Alignment Estimator)**.**

$$w_{align}(L,(P,T,F,M_0),M_F)(t) = |[(x,t) \in \mathit{align}((P,T,F,M_0),M_F,L)]| \\ \textit{where } t \in T$$

This algorithm only applies to Petri nets with at least one reachable final marking. The time complexity is $O(|T| \cdot |\mathit{align}(PN,M_F,L)|)$ plus the time to compute *align*. The alignment estimator has similarities with GDT_SPN discovery [130], which fits duration distributions to aligned logs. In our example in Figure 3.2, the trace $\langle a,b,b,d \rangle \in EL$ has one fitting path through the model involving the τ transition. The trace $\langle a,b,c,b,d \rangle \in EL$ does not fit the model EPN, as b is executed a second time and c is executed. Thus, alignments will (based on a cost function, or if that does not discriminate the options an arbitrary choice) include a log move on either b or a log move on c. If the alignments choose a b for a log move, then $w_{align}(EL,EPN,M_F,b) = 14$ and

$w_{align}(EL, EPN, M_F, \tau) = 2$. Alignments are not always deterministic, and consequently neither is w_{align}.

The design trade-offs between estimators are discussed in the context of results in Section 3.6.

3.4 Implementation and Evaluation

The six estimators introduced in Section 3.3 were implemented in the ProM framework [63][1]. For our evaluation, a discovery algorithm was applied to an event log. Where necessary, the result was converted to a Petri net. Each estimator was invoked on this Petri net, resulting in a SLPN. Finally, the conformance of the resulting SLPN was measured against the original log. For comparison, GDT_SPN discovery [130], an existing stochastic discovery algorithm, was also applied to the log, and the same conformance measures were applied. The implementation of this plugin in ProM 6.9 and later uses the Inductive Miner internally as an initial control flow discovery step, which has been updated from the gradient-descent procedure described in [130]. Externally, the implementation behaves like a direct discovery algorithm, while internally it is an estimator due to the use of a control-flow input from Inductive Miner. Six public reference event logs were used, and key characteristics are in Table 1.1. Algorithms, reference event logs and conformance measures are summarized as Figure 3.3.

Metrics chosen include all stochastic process conformance measures available at the time of the experiment. Truncated Earth Movers' Distance (EM) [95] provides a measure expressing the cost of transforming the distribution of activity traces from one stochastic language into another. We use a minimum probability mass parameter setting of 0.8 for feasibility. Entropy Precision (H_P) and Recall (H_F) [101] are stochastic conformance measures based on the entropy of equivalent automata constructed from a given log or model. Petri net entity count (places and transitions) and edge count are used as structural simplicity metrics, ensuring that conformance quality has not been achieved by sacrificing model simplicity and comprehensibility. Entity and arc counts have existing uses in process model evaluation [77, 90], and were preferred here over behavioural simplicity metrics [89], though these measures also have limitations, including specificity to Petri nets, and insensitivity to the stochastic perspective of SLPNs. The duration of a discovery process was also captured, and direct discovery times are compared with combined runtimes for discovery and estimation.

The experiments were run on a Windows 10 machine with 2.3GHz CPU and 50 Gb of memory allocated to each process on JDK 1.8.0_222. The results in this chapter are from experiments run in August 2023 on version 1.1.0 of the `spd_we` package. This version includes a fix to the implementation of w_{align} that could cause incorrect zero weighting of some silent transitions. Previously published experimental results from September 2020 included this bug. The full results for the experiments are available in Appendix B.

[1]Source code is accessible via `https://github.com/adamburkegh/spd_we`. See Appendix A.

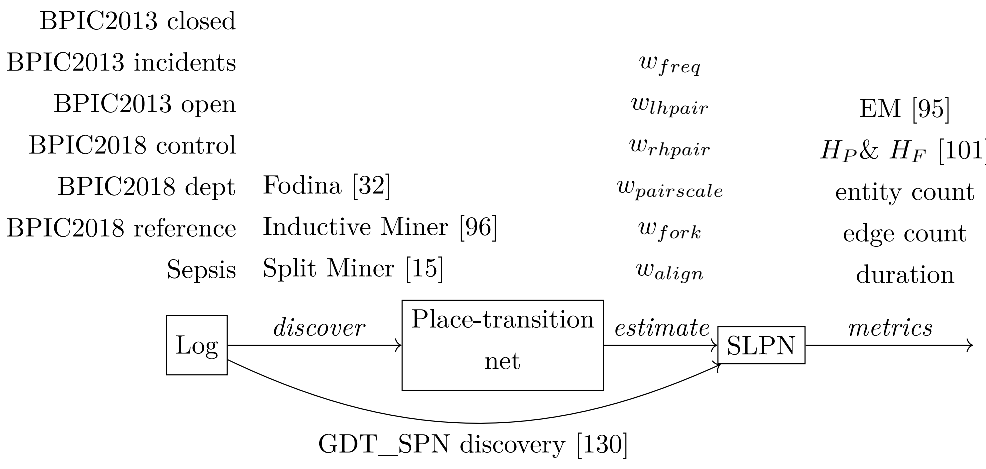


Figure 3.3: Estimator experimental evaluation design.

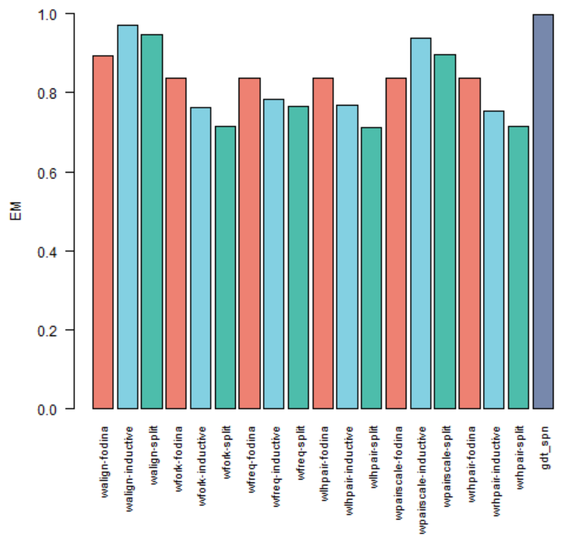
3.5 Results and Discussion

The estimators produced different, relevant, stochastic models when applied to a range of real-life logs. As seen in Figures 3.4 and 3.5, stochastic conformance for these models was comparable, but not uniformly better, than existing techniques, and was highly dependent on the discovery algorithm and log.

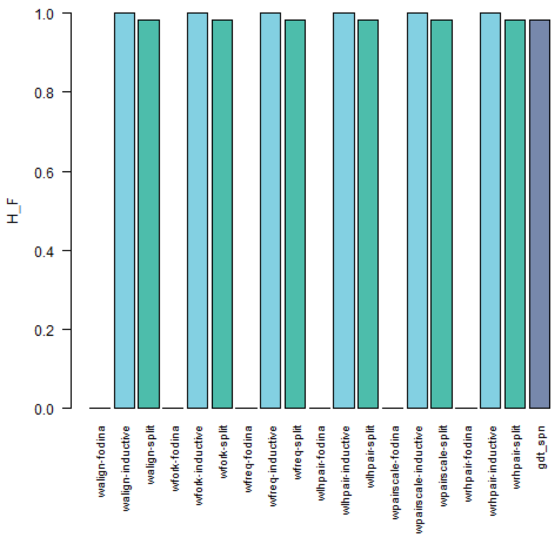
The estimators combined well with the Inductive Miner and Split Miner control discovery algorithms. When combined with the alignment estimator, w_{align}, either $w_{align} - inductive$ or $w_{align} - split$ showed the best Earth-Movers' Distance (EM) performance across BPIC2013 closed, BPIC2013 incidents, BPIC2018 Reference, and sepsis logs. Entropy recall (H_F) and precision (H_P) performance was also competitive on most logs, and for the BPIC2018 Control log in Figure 3.4, had the best precision performance. The alignment estimator is not the highest performing technique across every log and measure, however, as in the BPIC2018 Reference log in Figure 3.5, where the entropy precision performance is exceeded by the Mean-Scaled Activity-Pair Frequency Estimator, $w_{pairscale}$, and GDT_SPN discovery.

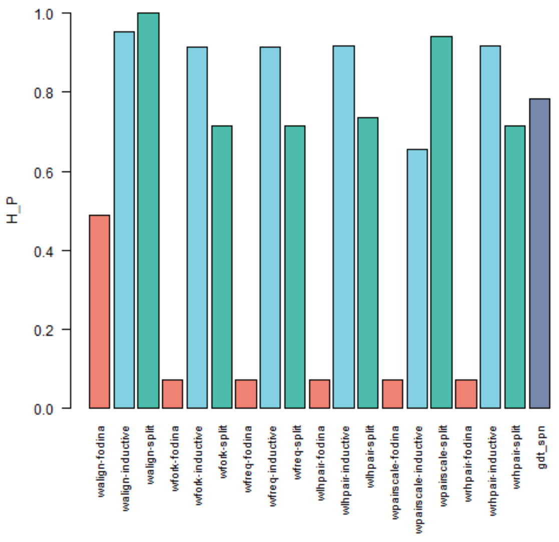
For the BPIC 2013 closed and incidents logs, Fodina returned a model without an initial place, to which estimators w_{fork} and w_{align}, and the EM, H_F and H_P measures do not apply. For some algorithm-estimator combinations, these conformance measures could not be calculated due to soundness, time or memory constraints. Nevertheless, in these results it is clear that using EM with probability mass coverage 0.8 is more sensitive to the stochastic perspective produced by estimators than the Entropy Precision and Recall measures.

Where GDT_SPN discovery [130] produced a model on which metrics could be calculated, the resulting models often conformed well to the logs, but were not consistently better than the estimator-produced models. There were a number of event logs where GDT_SPN discovery returned no model within the constraints of time (12 hour timeout) and machine memory, or where conformance measures were unable to be calculated within time (5 hour timeout) and memory constraints.



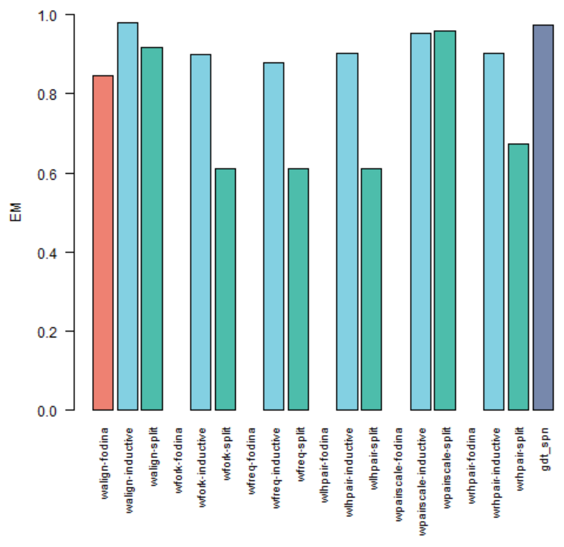
(a) earth movers' EM



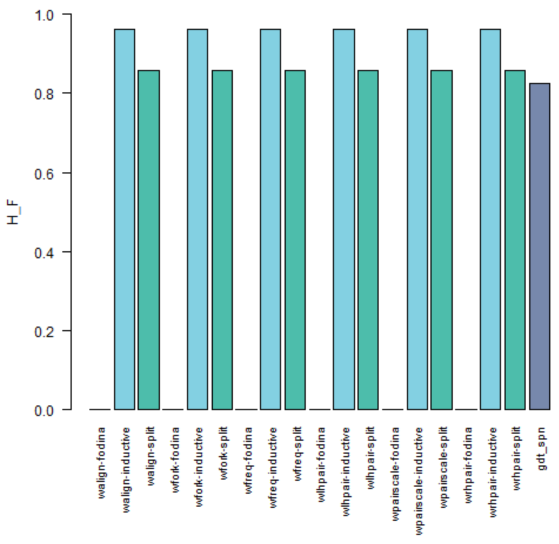


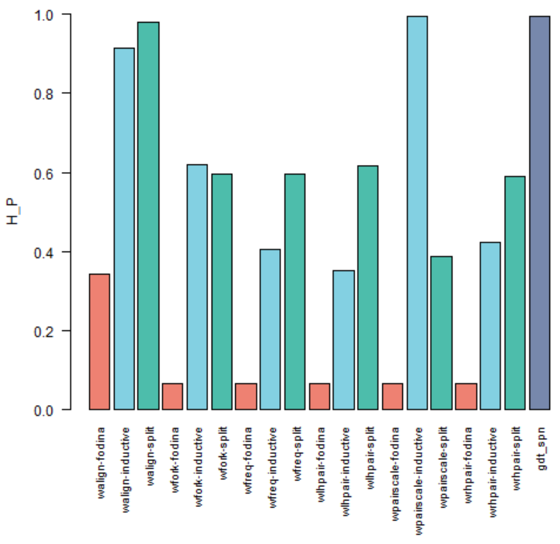
(b) entropy-recall H_F (c) entropy-precision H_P

Figure 3.4: Results on BPIC 2018 Control log categorized by {*estimator*}-{*control flow algorithm*}, plus GDT_SPN discovery.



(a) earth movers' EM





(b) entropy-recall H_F

(c) entropy-precision H_P

Figure 3.5: Results on BPIC 2018 Reference log.

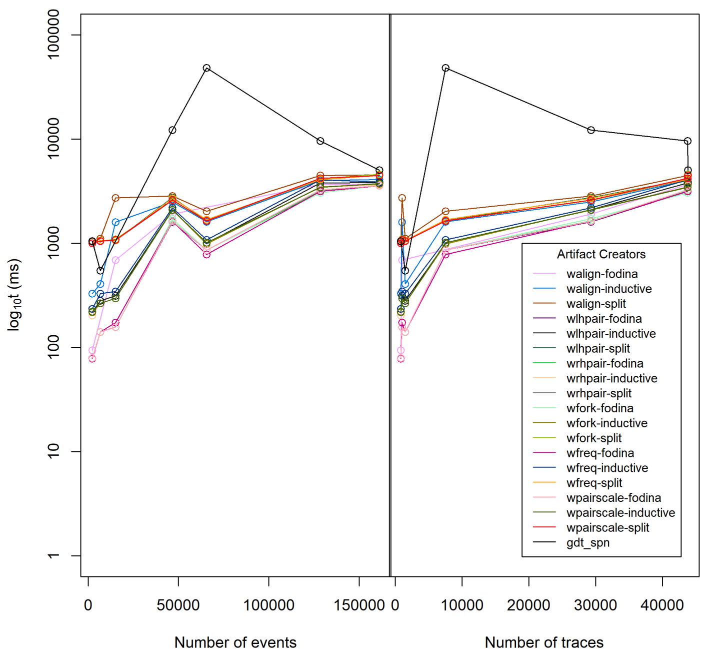


Figure 3.6: Run times for control flow discovery and weight estimation by event and trace count. 12 hour time out for GDT_SPN discovery [130] on sepsis log is excluded.

The run time of the estimators, which took never more than 10 seconds, was always comparable or better than GDT_SPN discovery, orders of magnitude better in some cases, as shown in Figure 3.6.

In summary, the new estimators, even the alignment-based w_{align}, are able to handle real-life event logs and outputs from existing discovery techniques much faster than existing approaches. Depending on the applied discovery technique, they can also achieve higher stochastic quality, providing alternatives to the GDT_SPN discovery technique when analyzing control flow and stochastic perspectives. The existence of multiple estimators, which combine with different control-flow algorithms, provide a number of alternative solutions when other discovery techniques return a poor quality output model, or no model at all.

Alternative approaches to designing estimators could take a more analytical starting point than this chapter. For example, given the probability density function for a particular process type, weight estimation could be treated as an optimisation problem where weights are solutions to a Maximum Likelihood Estimate (MLE). A strength of this approach would be theoretical support for use of a particular estimation function. Characterising the probability density function of a general process would itself be a research challenge: recent optimisation approaches to weight discovery [99, 112] result in functions which are non-linear, non-convex, and due to the use of alignments, non-deterministic.

3.6 Limitations

These experiments can be extended with larger logs containing more traces, events, and activities. However, even though our estimators returned results for each model and log combination quickly, the conformance metrics were the limiting factors in these experiments in terms of time and memory, which indicated that research was needed on towards more efficient stochastic conformance checking techniques. Some additional metrics have appeared in the scholarly literature since these experiments were performed, and they are incorporated in the experiments in Chapter 6.

The information used from the model varies across the estimators. The Frequency estimator, w_{freq}, ignores the structure of the model, and relies solely on activity frequency. Four of the estimators depend on direct-follow frequencies, with only basic interpolation of weights for silent transitions. Silent transitions can play structural roles in workflows, such as an optional activity, or the start of a block-structured subnet. These direct-follow estimators combined poorly with the Fodina discovery algorithm for some logs. This is at least partly due to Petri net representational bias in the presented framework. Fodina outputs a causal net, which was converted to a Petri net. The resulting Petri net includes a large number of silent transitions, often intermediating between transitions corresponding to activity pairs in the log. This can be seen distinctly in results for BPIC 2018 reference log in Figure 3.5, where w_{align} produces a stochastically useful model on the output of a Fodina input, but no other estimator does. For Split Miner and Inductive Miner, though they use other representations internally, the Petri net model produced used fewer silent transitions and were less impacted by this property.

The reliance on activity or direct-follow frequencies also limits estimator effectiveness in modelling the stochastics of concurrency. An estimator such as $w_{pairscale}$ uses the structure of the place-transition net, particularly double predecessor or double successor relations: for a transition t, it considers $(t\bullet)\bullet$ when calculating weights from pair frequencies. This will not identify concurrent transitions in a number of cases. In example Petri net model EPN in Figure 3.2, transitions labelled a and b can execute concurrently, and this is represented concisely. However, transitions a and b are not successors of one another, so $w_{pairscale}$ won't "credit" those transitions with observed $\langle a, b\rangle$ or $\langle b, a\rangle$ pair frequencies observed in traces in the log. All of the estimators apart from w_{align} have some variant of this limitation. To address it, estimators would need to use the structure of the place-transition's underlying reachability graph, rather than just the structure of the place-transition net itself.

The estimators, like many control-flow algorithms, also do not account for possible long-range dependencies, or make use of recent advances on this topic [110].

3.7 Summary

In this chapter we presented a framework for discovery of Stochastic Labelled Petri Nets (SLPNs) from logs. The framework leverages existing control flow discovery algorithms, and introduces *estimators* which transform discovered Petri nets into SLPNs. We introduced six estimators. Their implementations are publicly available, and evaluated against real-life logs using multiple stochastic conformance measures. The evaluation used three existing flow discovery algorithms, and an existing stochastic discovery technique, finding models of comparable quality, across a broader range of logs, in a generally shorter time.

The estimators presented here are not exhaustive, and we look forward to future research on novel, improved estimators. One extension since the publication of these results is an estimator for discovering Data Petri Nets from place-transition nets and event logs [112]. The estimator framework implies the possibility of "direct stochastic discovery" algorithms which do not use a separate control flow algorithm, but produce a control flow model as a side-effect of a stochastic one. The Toothpaste Miner family introduced in Chapter 4 and the State Snapshot Miner in Chapter 7 are two such techniques. A simplicity measure sensitive to both structural representation and stochastic information in a process model would be a useful evaluation tool for work in this area, and several are explored in Chapter 6.

Chapter 4

Discovery With Probabilistic Process Trees

[T]he artificial [has] two operative conditions — dependency and uncertainty.

Clive Dilnot
Desigining In The World of the Naturalised Artificial [60]

In this chapter, we introduce a family of related stochastic process discovery algorithms, which use a form of weighted process trees, called Probabilistic Process Trees (PPTs). We explore the trees' formal properties, semantics, and translation to a form of Stochastic Labelled Petri Nets (SLPNs) [95] (and Definition 8). The discovery algorithms, called the Toothpaste Miner framework, are introduced. These are based on reduction rules, and used to construct PPTs from event logs. An implementation of Toothpaste is shared, and evaluated on real-life logs, discovering high quality models across event logs from a variety of real-world domains.

As a running example, we consider a simplified claims process for an insurer, based on the teleclaims dataset[1]. There are four activities: *approve claim*, *reject claim*, *close claim*, and *advise claimant*, and the progress of a case is recorded as events by the insurer's software system. After process discovery on these events, we will have a PPT model that describes the ordering of activities, which activities are repeated, and which are mutually incompatible.

In this chapter, related work is surveyed in Section 4.1, and mathematical preliminaries are covered in Section 4.2, before formally introducing PPTs in Section 4.3. We discuss properties of PPTs and translation to a subset of SLPNs, as well as functions for determinism. A discovery algorithm framework, Toothpaste Miner, exploiting these techniques, is introduced in Section 4.4. Concrete reduction rules required for Toothpaste, and a normal form for PPTs, are detailed in Section 4.5. A Haskell implementation and experimental evaluation is covered in Section 4.6, with

[1]Teleclaims [7] is a realistic example event log `http://www.processmining.org/old-version/event-book.html`. See Table 1.1.

results in Section 4.7. Section 4.8 summarises the work in the chapter.

4.1 Related Work

This work provides an alternative to existing techniques to discover stochastic process models. In the process mining literature, a number of techniques leverage the success of control-flow discovery algorithms by adding stochastic information to a control-flow model [130, 86] (and Chapter 3). Other techniques use Markov Chain Monte Carlo (MCMC) [86] or distribution fitting with a goal of business process simulation [38]. The Toothpaste Miner introduced in Section 4.4 performs direct stochastic discovery, that is, discovery without a preceding control-flow discovery step.

The theory of probabilistic automata includes the formalisms of Stochastic Finite Automatons (SFAs) [152, 66] and Stochastic Deterministic Finite Automata (SDFAs) [152]. As the names suggest, these are state machines where transitions are governed by a probability function. Discovery algorithms for SDFA include the state merging algorithms ALERGIA [44] and MDL [147]. These work in polynomial time, and ALERGIA is competitive in real-world trials [150]. Toothpaste Miner uses a state merging strategy, but with an extension of process trees [6, p78-83], PPTs, rather than the prefix trees in ALERGIA. This allows output models that directly model concurrency and loops, and are more aligned with problems in process mining.

Process trees are models for processes which guarantee useful properties such as safeness and soundness, and are translatable to Petri nets. They are used by existing control-flow discovery techniques such as Inductive Miner [96] and genetic miners [34]. Similarly, PPTs are translatable to a form of weighted Petri net, discussed in Section 4.3. Region-based control-flow miners for Petri nets [42, 41], which use rule-based transformations, are another inspiration for our discovery technique. The MAXIMAL PATTERN MINER algorithm [106] is a region-based miner whose control-flow loop and concurrency identification rules served as a starting point for some rules in Section 4.5. Other rule sources are Petri net and SPN reductions [155, 146], and the Inductive Miner [97]. Our Toothpaste Miner algorithms assemble stochastic models with rich control-flow constructs in polynomial time, and maintain consistent quality across a variety of domain logs.

Though not a direct source for the rules in this chapter, the *structural foldings* of a GSPN performance model optimisation technique [136] have important parallels and some complementary strengths. The time dimension in GSPNs, relating to performance, is not represented in the models output by the Toothpaste Miner. Structural foldings identify patterns in Petri nets for four basic workflow constructs: sequence, choice, concurrency, and loops, which have corresponding process tree operators. Structural foldings, like Toothpaste rules, are reductions which always simplify the input net. Where Toothpaste Miner rules are classified and ranked qualitatively, in categories of determinism, impact on stochastic language, or quality metrics, structural foldings are constrained quantitatively in their impact on performance estimation error. As they are applied to a separate performance model whose starting point is the stochastic process model, and unlike a discovery technique, GSPN structural foldings can freely impact the activity or stochastic languages of the process.

Our experimental evaluation in Section 4.6.2 makes use of quality metrics from the literature.

The Earth-Movers' Distance (EMD) measure [95] represents shared probability mass between models or between logs and models, building on the well-known Levenshtein string edit distance. Measures for entropy precision and recall (fitness) [101], and entropic relevance [14] have been defined using Stochastic Deterministic Finite Automata [152]. The links between simplicity and entropy have also been investigated [89], emphasizing a distinction between *behavioural* simplicity and *structural* simplicity. An Alpha-precision measure [59] interprets precision in terms of the statistical significance of different paths in comparing different models and logs. Many of these measures depend on trace probability (Definition 13). Earth-movers distance [95] and a variant of Alpha-precision [59] were used for the evaluation in this chapter.

4.2 Preliminaries

We start by expanding on Definition 10 for Stochastic Finite Automata (SFA).

4.2.1 Weighted Automata

One useful class of SFAs are those with a single terminal state. For later use, we associate arcs with weights, from which SFA probabilities can be straightforwardly calculated.

Definition 21 (Weighted Stochastic Finite Automata)**.** *A* Weighted Stochastic Finite Automaton *(WSFA) is a five-tuple* $(S, Act, \hookrightarrow, s_0, s_\omega)$*, where*

- $\hookrightarrow \subset S \times Act \times S \to \mathbb{R}^+$ *is a weight function,*
- $GS(s_1, x, s_2) = \frac{w}{W_T} \equiv \hookrightarrow (s_1, x, s_2) = w$
  where $W_T = \sum_{s_3 \in S, y \in Act} \hookrightarrow (s_1, y, s_3)$ *where* $\hookrightarrow$ *is defined,*
- (S, Act, GS, s_0) *is an SFA, and*
- s_0 *has no incoming arcs*
- $s_\omega \in S$ *is the final state, and a deadlock state, with no outgoing arcs.*

For notational convenience, $\hookrightarrow (s_1, x, s_2) = w$ is written as $s_1 \overset{x[w]}{\hookrightarrow} s_2$. The set of all WSFAs is $\mathcal{WS}$. The sequence of symbols generated by proceeding from the initial state form a path. Traces generated by the model are finite sequences of activities on paths which include the final state.

$s_1 \overset{x[w]}{\hookrightarrow} s_2$ can also be used to indicate an instance member of the transition set. Figure 4.1 shows a simple example WSFA, E_{tdrive}, with a choice between activities *walk* and *drive*. $\hookrightarrow = \{s_0 \overset{walk[7]}{\hookrightarrow} s_\omega, s_0 \overset{drive[3]}{\hookrightarrow} s_\omega\}$, meaning there is a $\frac{7}{10}$ chance of walking and a $\frac{3}{10}$ chance of driving.

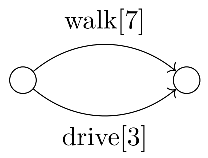


Figure 4.1: Example WSFA model for travel choices, E_{tdrive}.

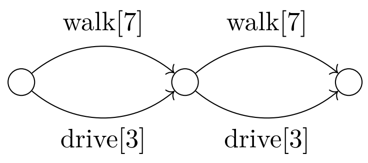


Figure 4.2: Two sequential trips, $E_{tdrive} +\!\!+ E_{tdrive}$.

4.2.2 Operations on Weighted Automata

Two SFAs $(S_1, Act_1, GS_1, s_{1.0})$, $(S_2, Act_2, GS_2, s_{2.0})$ are *disjoint* when $S_1 \cap S_2 = \varnothing$.

When defining these operators, we follow the convention that states in operand automata are disjoint [133], as it is straightforward to construct a copy function to ensure it.

We define concatenation, union and race operators on weighted automata.

Concatenation ($+\!\!+$) chains two automata together.

Definition 22 (WSFA concatenation). *Two automatons execute serially. Let $E_1 = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega})$ and $E_2 = (S_2, A_2, \hookrightarrow_2, s_{2.0}, s_{2.\omega})$ be WSFAs. Further let E_1 and E_2 be WSFA already made disjoint through copying if necessary. Then $E_1 +\!\!+ E_2 = (S^{+\!\!+}, A^{+\!\!+}, \hookrightarrow^{+\!\!+}, s_0^{+\!\!+}, s_\omega^{+\!\!+})$, with:*

$$
\begin{aligned}
S^{+\!\!+} &= S_1 \cup S_2 \cup \{(s_{1.\omega}, s_{2.0})\} \backslash \{s_{1.\omega}, s_{2.0}\} \\
A^{+\!\!+} &= A_1 \cup A_2 \\
\hookrightarrow^{+\!\!+} &= \hookrightarrow_1 \cup \hookrightarrow_2 \\
&\quad \cup \{s_1 \overset{x[w]}{\hookrightarrow} (s_{1.\omega}, s_{2.0}) \mid s_1 \overset{x[w]}{\hookrightarrow}_1 s_{1.\omega}\} \\
&\quad \cup \{(s_{1.\omega}, s_{2.0}) \overset{x[w]}{\hookrightarrow} s_2 \mid s_{2.0} \overset{x[w]}{\hookrightarrow}_2 s_2\} \\
&\quad \backslash \{s \overset{x[w]}{\hookrightarrow} s' \mid s = s_{2.0} \vee s' = s_{1.\omega}\} \\
s_0^{+\!\!+} &= s_{1.0} \\
s_\omega^{+\!\!+} &= s_{2.\omega}
\end{aligned}
$$

The disjointness condition ensures that no states are shared between operands in operations like $E +\!\!+ E$. A new shared state, the tuple $(s_{1.\omega}, s_{2.0})$, replaces the final state of the left operand and the initial state of the right operand. WSFA concatenation is associative, allowing extension to a multi-child operator by composition.

Figure 4.2 shows the automata for two sequential trips, $E_{tdrive} +\!\!+ E_{tdrive}$.

A weighted disjoint union operator $\sqcup$ allows exactly one of its operands to completely execute. A *union on WSFA* $\sqcup$ constructs a new automaton that may behave as either of its component WSFAs, with a probability determined by the relative weights of the existing arcs.

Definition 23 (WSFA union). *Let $E_1 = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega})$ and $E_2 = (S_2, A_2, \hookrightarrow_2, s_{2.0}, s_{2.\omega})$*

be WSFAs already made disjoint through copying. Then $E_1 \sqcup E_2 = (S^{\sqcup}, A^{\sqcup}, \hookrightarrow^{\sqcup}, s_0^{\sqcup}, s_\omega^{\sqcup})$*, with:*

$$\begin{aligned}
S^{\sqcup} &= S_1 \cup S_2 \cup \{(s_{1.0}, s_{2.0}), (s_{1.\omega}, s_{2.\omega})\} \backslash \{s_{1.0}, s_{2.0}, s_{1.\omega}, s_{2.\omega}\} \\
A^{\sqcup} &= A_1 \cup A_2 \\
\hookrightarrow^{\sqcup} &= \hookrightarrow_1 \cup \hookrightarrow_2 \\
&\quad \cup \{(s_{1.0}, s_{2.0}) \overset{x[w]}{\hookrightarrow} s \mid s_{1.0} \overset{x[w]}{\hookrightarrow}_1 s \vee s_{2.0} \overset{x[w]}{\hookrightarrow}_2 s\} \\
&\quad \cup \{s \overset{x[w]}{\hookrightarrow} (s_{1.\omega}, s_{2.\omega}) \mid s \overset{x[w]}{\hookrightarrow}_1 s_{1.\omega} \vee s_{2.0} \overset{x[w]}{\hookrightarrow}_2 s_{2.\omega}\} \\
&\quad \backslash \{s \overset{x[w]}{\hookrightarrow} s' \mid s \in \{s_{1.0}, s_{2.0}\} \vee s' \in \{s_{1.\omega}, s_{2.\omega}\}\} \\
s_0^{\sqcup} &= (s_{1.0}, s_{2.0}) \\
s_\omega^{\sqcup} &= (s_{1.\omega}, s_{2.\omega})
\end{aligned}$$

As a gloss, the operator merges the initial and final states of the original automata, which establishes a weighted choice over which automata to execute.

In a weighted race, represented by $||$, two automatons progress in parallel. At each step, the child automaton to progress next is determined by a weighted choice between arcs. The result is a Cartesian product of possible states, and arcs between them.

Definition 24 (WSFA race)**.** *Let* $E_1 = (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega})$ *and* $E_2 = (S_2, A_2, \hookrightarrow_2, s_{2.0}, s_{2.\omega})$ *be WSFAs already made disjoint through copying. Then* $E_1 \,||\, E_2 = (S^{||}, A^{||}, \hookrightarrow^{||}, s_0^{||}, s_\omega^{||})$*, with:*

$$\begin{aligned}
S^{||} &= \{(s_1, s_2) \mid s_1 \in S_1 \wedge s_2 \in S_2\} \\
A^{||} &= A_1 \cup A_2 \\
\hookrightarrow^{||} &= \{(s_{1.y}, s_{2.y}) \overset{x[w]}{\hookrightarrow} (s_{1.z}, s_{2.y}) \mid s_{1.y} \overset{x[w]}{\hookrightarrow}_1 s_{1.z}\} \\
&\quad \cup \{(s_{1.y}, s_{2.y}) \overset{x[w]}{\hookrightarrow} (s_{1.y}, s_{2.z}) \mid s_{2.y} \overset{x[w]}{\hookrightarrow}_2 s_{2.z}\} \\
s_0^{||} &= (s_{1.0}, s_{2.0}) \\
s_\omega^{||} &= (s_{1.\omega}, s_{2.\omega})
\end{aligned}$$

In a race between WSFAs, there is no "communication" between states in the sense used by process algebras (synchronous shared events).

Figure 4.3 shows the automata for two trips conducted in a race (say between two travellers). A second traveller chooses between walking or taking the train:

$$E_{ttrain} = (\{s_0, s_\omega, \{walk, train\}, \{s_0 \overset{walk[2]}{\hookrightarrow} s_\omega, s_0 \overset{train[8]}{\hookrightarrow} s_\omega\}, s_0, s_\omega)$$

The resulting WSFA $E_{tdrive} \,||\, E_{ttrain}$ simulates both automata executing at once, with each transition progressing one automaton or the other.

WSFA concatenation, union and race relations are associative, allowing extension to multi-child operators by composition. Union and race are also commutative.

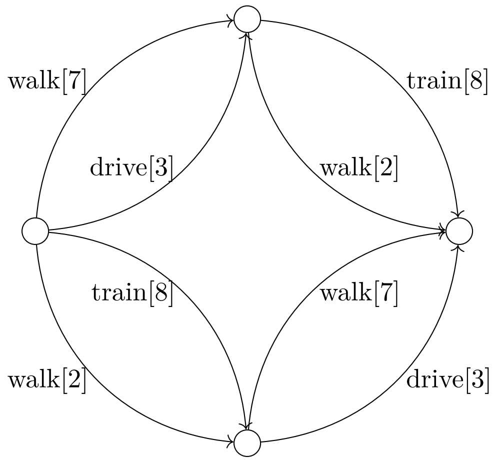


Figure 4.3: Two trips executed in a race, $E_{tdrive} \,||\, E_{ttrain}$.

4.3 Probabilistic Process Trees (PPTs)

Probabilistic Process Trees are stochastic finite automata where probabilities are derived from weights on each node, which have to be balanced across and down the tree in certain consistent ways. In this section, we first formally define PPTs. Secondly we give a translation to SLPNs. Next we discuss challenges for PPT to SLPN translation. Lastly we introduce functions for determinism of a tree.

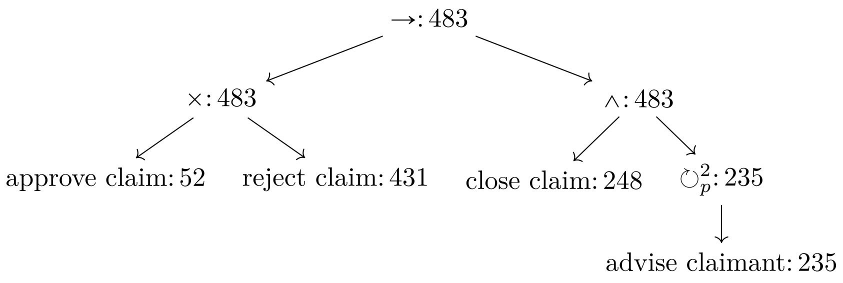


Figure 4.4: Detail of a claims process as a Probabilistic Process Tree (PPT).

4.3.1 Tree and Operator Definitions

We formally define first the structure of PPTs and then each constituent operator.

Definition 25 (Probabilistic Process Trees (PPTs))**.** *Let $x: w$ be a node, where $w \in \mathbb{R}^+$ is a weight and x the remainder. The universe of PPTs is $\mathcal{PPT}$, recursively and exhaustively defined as:*

1. *A single activity. For $a \in \mathcal{A}$, $a: w \in \mathcal{PPT}$.*

2. *A silent activity, represented by the constant τ. Note $\tau \notin \mathcal{A} \wedge \tau: w \in \mathcal{PPT}$.*

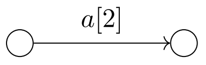
3. *A unary operator $\oplus_1$. Given $u \in \mathcal{PPT}$, then $\oplus_1(u): w \in \mathcal{PPT}$.*

4. *An n-ary operator $\oplus_n$ over one or more child trees. Given $m \geqslant 1$, $u_1, ..., u_m \in \mathcal{PPT}$, then $\oplus_n(u_1, ..., u_m)\colon w \in \mathcal{PPT}$.*

The PPT operators are $\bigoplus = \{\rightarrow, \wedge, \times, \circlearrowleft_n, \circlearrowleft_p\}$: sequence, concurrency, choice, a fixed loop, and a probabilistic loop. These are related to control-flow process tree operators [97], but include weight semantics. These semantics are described by an equivalent WSFA for each node, which recursively defines the semantics function $\mathbf{wa} : \mathcal{PPT} \rightarrow \mathcal{WS}$.

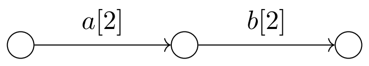
Leaf nodes.

$$\mathbf{wa}(x\colon w) = (\{s_0, s_\omega\}, \{x\}, \{s_0 \overset{x[w]}{\hookrightarrow} s_\omega\}, s_0, s_\omega) \text{ for } x \in \mathcal{A} \cup \{\tau\}$$

Example: $\mathbf{wa}(a\colon 2) =$ 

Sequential operator. The $\rightarrow$ sequential operator executes its children in order. As a sequence specifies that every subtree executes, the weight of children is the parent's weight.

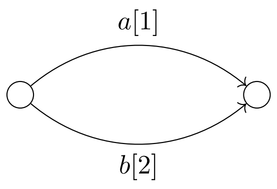
$$\forall \rightarrow(x_1\colon w_1, x_2\colon w_2)\colon w \blacksquare w_1 = w_2 = w \wedge$$
$$\mathbf{wa}(\rightarrow(x_1\colon w, x_2\colon w)\colon w) = \mathbf{wa}(x_1\colon w_1) \mathbin{+\!\!+} \mathbf{wa}(x_2\colon w_2)$$

$$\text{Example: } \mathbf{wa}\Bigg(\begin{tikzcd}[column sep=tiny] & \rightarrow\colon 2 \arrow[dl] \arrow[dr] & \\ \text{a}\colon 2 & & \text{b}\colon 2\end{tikzcd}\Bigg) =$$


The sequential operator is associative, so the binary operator can be extended to a multi-child operator by composition. In the remainder of this thesis, we may omit the child weights for sequences, writing $\rightarrow(x, y)\colon w$ rather than $\rightarrow(x\colon w, y\colon w)\colon w$.

Choice operator. The $\times$ operator is an exclusive choice between its children.

$$\forall \times(x_1\colon w_1, x_2\colon w_2)\colon w \blacksquare w = \sum_{i=1}^{i=m} w_i \wedge$$
$$\mathbf{wa}(\times(x_1\colon w_1, x_2\colon w_2)\colon w) = \mathbf{wa}(x_1\colon w_1) \sqcup \mathbf{wa}(x_2\colon w_2)$$

$$\text{Example: } \mathbf{wa}\Bigg(\begin{tikzcd}[column sep=tiny] & \times\colon 3 \arrow[dl] \arrow[dr] & \\ \text{a}\colon 1 & & \text{b}\colon 2\end{tikzcd}\Bigg) =$$


The choice operator is associative and commutative, so the binary operator can be extended to a multi-child operator by composition.

Concurrency operator. The $\wedge$ operator represents parallel execution of child trees. Each child process participates in a weighted race with its siblings.

$$\forall \wedge(x_1\colon w_1, ..., x_m\colon w_m)\colon \blacksquare w = \sum_{i=1}^{i=m} w_i \wedge$$

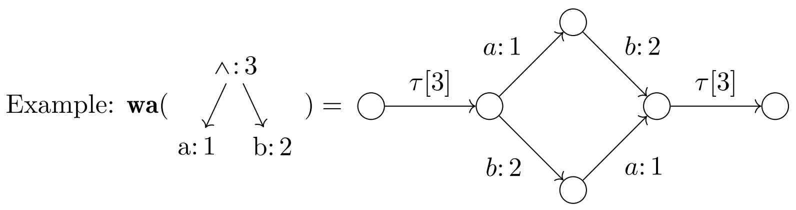
$$\mathbf{wa}(\wedge(x_1\colon w_1, ..., x_m\colon w_m)\colon w) = (S^\wedge, A^\wedge, \hookrightarrow^\wedge, s_0, s_\omega)$$

$$\text{where } (S_r, A_r, \hookrightarrow_r, s_{r.0}, s_{r.\omega}) = x_1\colon w_1 \,||\, ... \,||\, x_m\colon w_m$$

$$S^\wedge = S_r \cup \{s_0, s_\omega\}$$

$$A^\wedge = A_r \cup \{\tau\}$$

$$\hookrightarrow^\wedge \;=\; \hookrightarrow_r \cup \{s_0 \overset{\tau[w]}{\hookrightarrow} s_{r.0}, s_{r.\omega} \overset{\tau[w]}{\hookrightarrow} s_\omega\}$$

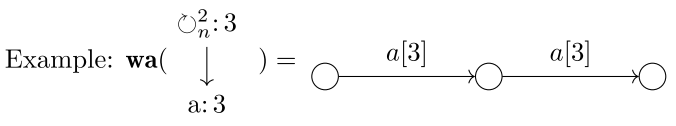


The concurrency operator $\wedge$ is commutative but not associative. This is discussed further, with counter-examples, in Section 4.3.3. The additional silent entry and exit states make translation to block-structured SLPNs more straightforward.

Fixed loops. The $\circlearrowright_n^m$ operator repeats its child m times, where $m \in \mathbb{N}$.

$$\circlearrowright_n^m(u)\colon w \equiv \rightarrow(u, \;...\; m \text{ times } ...)\colon w$$

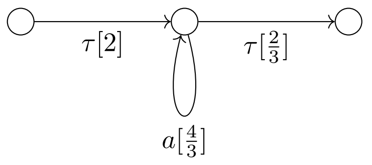
$$\forall \circlearrowright_n^m(x\colon w_1)\colon w_2 \blacksquare w_1 = w_2$$



Fixed loops are a largely syntactical shorthand for sequences which repeat the same element. As for sequences, the weight of the fixed loop is the same as the weight of its children.

Probabilistic Loops. The probabilistic loop operator $\circlearrowright_p^\rho(u)\colon w$ executes a single child tree with a probability of exiting at each iteration of $\frac{1}{\rho}$ where $\rho \in \mathbb{R}^+ \wedge \rho > 1$. The weight of the child node is the weight of the parent loop. The corresponding trace language is infinite, though the number of states remains finite.

$$\begin{aligned}
&\mathbf{wa}(\circlearrowleft_p^{\rho}(x\colon w)\colon w) = (S_L, A_L, \hookrightarrow_L, s_0, s_\omega)\\
\text{where } &S_L = S_1 \cup \{s_0, s_\omega\}\\
&A_L = A_1 \cup \{\tau\}\\
&\hookrightarrow_L \;= \{s_0 \overset{\tau[w]}{\hookrightarrow} s_{1.0}\} \cup \{s_{1.0} \overset{\tau[\frac{w}{\rho}]}{\hookrightarrow} s_\omega\}\\
&\quad \cup \{s_{1.1} \overset{x[v]}{\hookrightarrow} s_{1.2} \mid s_{1.1} \overset{x[v']}{\hookrightarrow}_1 s_{1.2} \wedge s_{1.2} \neq s_{1.\omega} \wedge v = \frac{v' \cdot (\rho - 1)}{\rho}\}\\
&\quad \cup \{s_{1.1} \overset{x[v]}{\hookrightarrow} s_{1.0} \mid s_{1.1} \overset{x[v']}{\hookrightarrow}_1 s_{1.\omega} \wedge v = \frac{v' \cdot (\rho - 1)}{\rho}\}
\end{aligned}$$

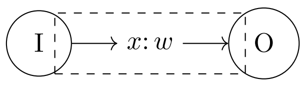
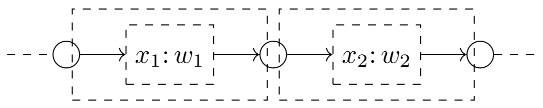
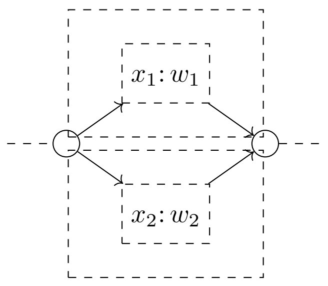
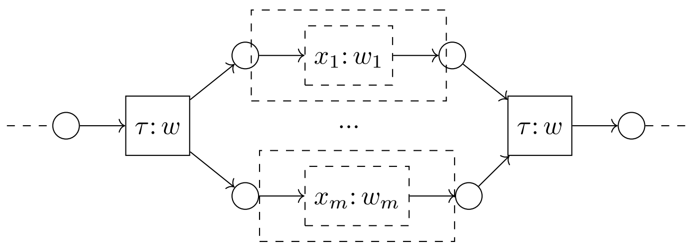
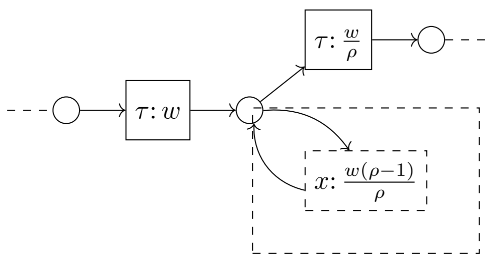
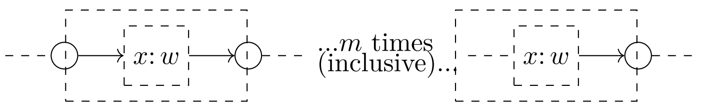
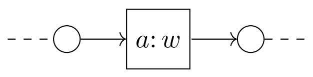
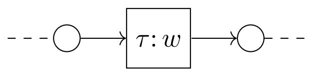
Example: $\mathbf{wa}\left(\begin{array}{c}\circlearrowleft_p^3\colon 2\\ \downarrow\\ \text{a}\colon 2\end{array}\right) =$ 

In the WSFA, the initial and final states of the child WSFA merge, and loop and exit arcs are added with weights matching the probability of continuation and exit respectively. Those parts of the WSFA corresponding to the child are scaled by continuation probability. As with sequences, we may omit writing the child weights, as by definition the weight of the loop child is the same as the loop operator.

Having described all of the operators, this concludes Definition 25, formally describing PPTs. The number of nodes in a PPT $u \in \mathcal{PPT}$ is given by $|u|$. When PPT u_1 is a subtree of PPT u_2, we write $u_1 \hat{\subseteq} u_2$. Subtrees can be any node under the parent, not just the immediate children. For PPTs, we define $=$ as all operators, children and weights being equal and in the same order. Throughout this thesis, we will continue the convention that u variables represent entire PPTs including weights, and $x\colon w$ nodes state weights explicitly.

Figure 4.4 shows a PPT modelling our running example, which describes both variations and probabilities in a simple insurance claim process.

Table 4.1: Translation of PPTs to SLPNs.

Probabilistic Process Tree	SLPN
$x\colon w$ (root)	
$\rightarrow\colon w$ $\swarrow$ $\searrow$ $x_1\colon w_1$ $x_2\colon w_2$	
$\times\colon w$ $\swarrow$ $\searrow$ $x_1\colon w_1$ $x_2\colon w_2$	
$\wedge\colon w$ $\swarrow$ ... $\searrow$ $x_1\colon w_1$ $x_m\colon w_m$	 $\tau\colon w$ $x_1\colon w_1$ ... $x_m\colon w_m$ $\tau\colon w$
$\circlearrowleft_p^{\rho}\colon w$ $\downarrow$ $x\colon w$	 $\tau\colon w$ $\tau\colon \frac{w}{\rho}$ $x\colon \frac{w(\rho-1)}{\rho}$
$\circlearrowleft_n^{m}\colon w$ $\downarrow$ $x\colon w$	
$a\colon w$	
$\tau\colon w$	

4.3.2 Petri Net Translation

Stochastic Labelled Petri Nets (SLPNs) are a type of weighted Petri nets specified in Definition 8. Probabilistic Process Trees can be translated to SLPNs with the same semantics, connecting them to both the formal results available from the study of Petri nets, and the existing process mining tool infrastructure. The translation scheme is shown in Table 4.1.

SLPN Translation

In Table 4.1 we show a translation scheme from PPTs to SLPNs; though others may be possible, this is the translation scheme used through the thesis. Each PPT construct on the left translates to an SLPN construct on the right. Dashed lines for PPTs indicate the child tree may be any arbitrary PPT satisfying weight rules, say, $x\colon w$. The dashed blocks in the corresponding SLPN represent the result of translating this child PPT into an SLPN block. Solid lines, as for the silent transitions introduced for loops or concurrency, indicate concrete structural SLPN elements.

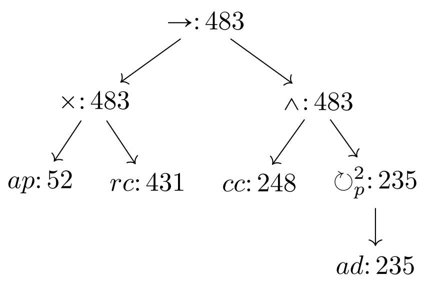
Every translated SLPN has an initial place and a final place. These are created for the root of the tree and then provided to the child translation. The remaining translations all expect an initial and final place as input. The sequence operator $\rightarrow$ translates to a chain of nets, where the final place of the first becomes the initial place of the second. The choice operator $\times$ translates to two subnets sharing initial and final places only.

The concurrency operator $\wedge$ translates to a concurrent block between opening and closing silent transitions. Each child tree becomes its own parallel block between new input and output places. The m possible children are shown, as the concurrency operator is not associative. The resulting net describes a concurrent race between the subnets for each translated child.

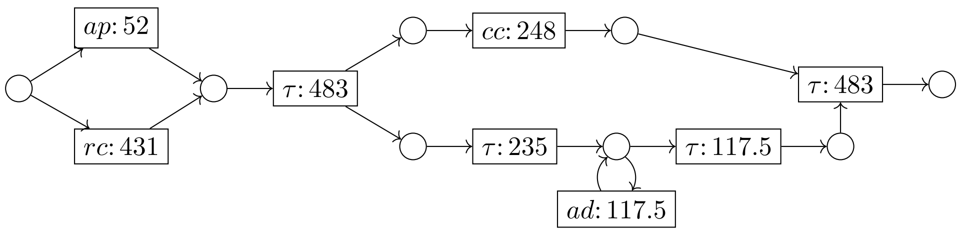
In the PPT probabilistic loop $\circlearrowleft_p^\rho$, each iteration is a Bernoulli trial, resulting in a the number of iterations being a geometric distribution. Each iteration of the loop is reflected as a weighted choice between exit or continuation. Recalling that Pr is a probability calculation,

$$\begin{aligned}
Pr(\circlearrowleft_p \text{ exit}) &= \frac{\text{exit transition weight}}{\sum w_i} \\
\sum w_i &= \frac{w(\rho-1)}{\rho} + \frac{w}{\rho} = w \\
Pr(\circlearrowleft_p \text{ exit}) &= \frac{w}{\rho}\frac{1}{w} = \frac{1}{\rho} \text{ , as in definition of } \circlearrowleft_p \\
Pr(\circlearrowleft_p \text{ continuation}) &= \frac{\text{continuation transition weight}}{\sum w_i} = \frac{w(\rho-1)}{\rho}\frac{1}{w} = \frac{\rho-1}{\rho}
\end{aligned}$$

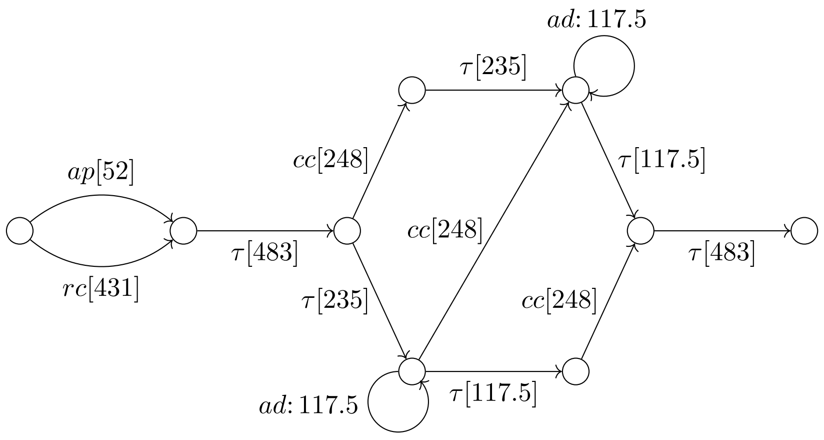
Fixed loops $\circlearrowleft_n^m$ translate as for sequences, chaining m blocks of the same child node. Lastly, leaf nodes $a\colon w$ and $\tau\colon w$ are represented as concrete transitions between the provided input and output places.



(a) PPT.



(b) SLPN.



(c) WSFA

Figure 4.5: PPT translation example to SLPN and equivalent WSFA, giving the semantics.

Figure 4.5 shows an example PPT translated to an SLPN, and the equivalent WSFA.

PPT and SLPN Non-equivalence

Previous work has shown challenges establishing the stochastic language for SLPNs which combine silent transitions and loops [102], as in Figure 4.6. No PPT translates directly to this net, though it may be possible to construct a PPT with the equivalent stochastic language. The probabilistic loop ($\circlearrowright_p$) body in PPTs is contained within a block with one entry and one exit, but in Figure 4.6 the loop has two separate exit points through b and c. Another simple example is an SLPN with two final (output) states, as in Figure 4.7; by the root translation rule in Table 4.1, all PPT SLPNs only have one.

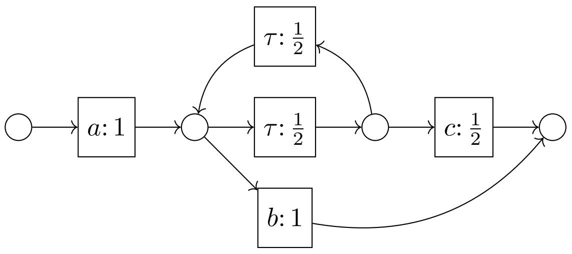


Figure 4.6: SLPN. The PPT to SLPN translation cannot produce this net.

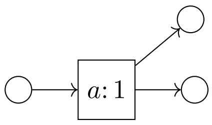


Figure 4.7: An SLPN which cannot be generated by the PPT root rule.

4.3.3 Translation Challenges

Translating stochastic models of one type into another presents a number of challenges. This section discusses constraints that shape the design of PPT to SLPN translation, particularly around the interaction of stochastics, silence and concurrency. In a number of cases, control-flow languages can be equivalent where stochastic languages are not. These are not cases of incomplete translation from PPTs to SLPNs, but where certain PPT constructs are not equivalent to one another, unlike their control-flow analogues.

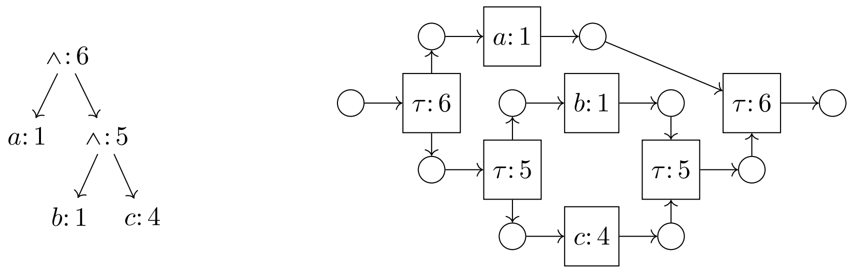
Definition 26 (Concurrent and Serial Trees)**.** *A tree parented by concurrency operator* $\wedge$ *is called a* concurrent tree*. Trees and subtrees with no concurrent ancestor are* serial*.*

Two cases which have no equivalent for control-flow models concern nesting concurrency operators and silent transitions in sequence. In both cases stochastic languages differ while trace languages remain the same.

Nesting Concurrency Operators

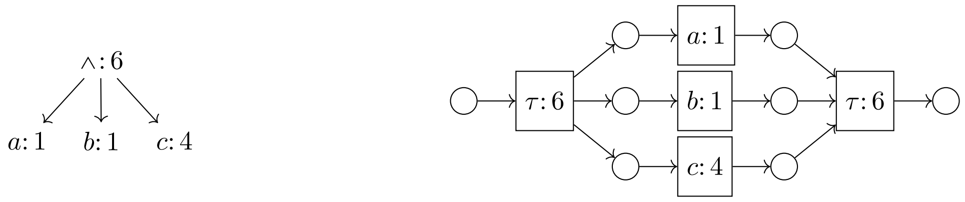
Nesting concurrent operators cannot be collapsed into a single operator without causing information loss which impacts the stochastic language. Consider the two trees in Figure 4.8. Using SLPN semantics, a silent transition is used to initiate concurrency. The transition $\tau\colon 5$ may execute before the $a\colon 1$ transition but still have the a transition execute before b or c. The trace $\langle a, b, c\rangle$ is then supported by two paths in the net: $\langle \tau\colon 6, a\colon 1, \tau\colon 5, b\colon 1, c\colon 4, \tau\colon 6\rangle$ and $\langle \tau\colon 6, \tau\colon 5, a\colon 1, b\colon 1, c\colon 4, \tau\colon 6\rangle$. This results in a difference in probability for $\langle a, b, c\rangle$ across the models in Figures 4.8a and 4.8b, and shows that $\wedge(a\colon 1, \wedge(b\colon 1, c\colon 4)\colon 6$ has a different stochastic language to $\wedge(a\colon 1, b\colon 1, c\colon 4)\colon 6$.

Showing the calculation for trace $\langle a, b, c\rangle$ in model 4.8a, the probability is the sum of the two



$$\{\langle a,b,c\rangle^{\frac{11}{180}}, \langle a,c,b\rangle^{\frac{44}{180}}, \langle b,a,c\rangle^{\frac{1}{36}}, \langle b,c,a\rangle^{\frac{1}{9}}, \langle c,a,b\rangle^{\frac{5}{18}}, \langle c,b,a\rangle^{\frac{5}{18}}\}$$

(a) Nested concurrency operator ∧ in PPT and SLPN form, and the stochastic language.



$$[\langle a,b,c\rangle^{\frac{1}{30}}, \langle a,c,b\rangle^{\frac{2}{15}}, \langle b,a,c\rangle^{\frac{1}{30}}, \langle b,c,a\rangle^{\frac{2}{15}}, \langle c,a,b\rangle^{\frac{1}{3}}, \langle c,b,a\rangle^{\frac{1}{3}}]$$

(b) Flattened concurrency operator ∧ in PPT and SLPN form, and the stochastic language.

Figure 4.8: Example SLPNs where nesting concurrency operators do not have the same stochastic language as a single concurrent parent operator.

possible path probabilities.

$$\begin{aligned} Pr(\text{path } \langle \tau\colon 6, a\colon 1, \tau\colon 5, b\colon 1, c\colon 4, \tau\colon 6\rangle) &= \frac{1}{6}\cdot\frac{1}{5} = \frac{1}{30} \\ Pr(\text{path } \langle \tau\colon 6, \tau\colon 5, a\colon 1, b\colon 1, c\colon 4, \tau\colon 6\rangle) &= \frac{5}{6}\cdot\frac{1}{6}\cdot\frac{1}{5} = \frac{1}{36} \\ Pr(\text{trace } \langle a,b,c\rangle) &= \frac{11}{180} \end{aligned}$$

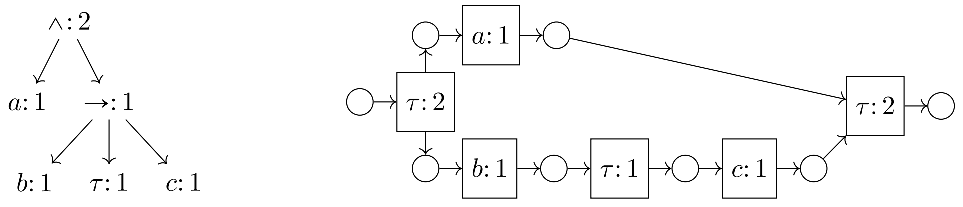
By contrast, in model 4.8b, only one path is possible

$$Pr(\text{path } \langle \tau\colon 6, a\colon 1, b\colon 1, c\colon 4, \tau\colon 6\rangle) = \frac{1}{6}\cdot\frac{1}{5} = \frac{1}{30}$$

Accordingly, an SLPN-equivalent concurrency operator is not nestable while maintaining the stochastic language. Furthermore, the concurrency operator is not associative, as $\wedge(a\colon 1, \wedge(b\colon 1, c\colon 4)\colon 6$ has a different stochastic language to $\wedge(\wedge(a\colon 1, b\colon 1), c\colon 4)\colon 6$.

Silent Activities in Concurrent Trees

Silent transitions in sequence do not affect trace languages, but can affect stochastic languages when within concurrent trees. Figure 4.9 shows the impact of a concurrency operator on silent transitions in sequence. Without the silent transition $\tau\colon 1$, the trace $\langle b,a,c\rangle$ would have probability $\frac{1}{4}$. With the silent transition in the sequence, it has probability $\frac{3}{8}$, due to the cumulative probability of two execution orders $\langle \tau\colon 2, b\colon 1, a\colon 1, \tau\colon 1, c\colon 1, \tau\colon 2\rangle$ and $\langle \tau\colon 2, b\colon 1, \tau\colon 1, a\colon 1, c\colon 1, \tau\colon 2\rangle$.



$$[\langle a,b,c\rangle^{\frac{1}{2}}, \langle b,a,c\rangle^{\frac{3}{8}}, \langle b,c,a\rangle^{\frac{1}{8}}]$$

Figure 4.9: Concurrency operator changing the weight impact of silence.

As we saw with silent transitions in sequence, the paths for a subtree in a concurrent tree may require information not local to the subtree, but based on other nodes under a (grand-)parent concurrent tree. It is the only operator which "interjects" activities into subtree paths, and this behaviour is not a property of serial trees.

4.3.4 Determinism

Determinism is the property where the next activity in a trace can always be reached by only a single path through a process tree.

Definition 27 (Deterministic Probabilistic Process Tree (DPPT))**.** *Let $u \in \mathcal{PPT}$ with activities $A \subseteq \mathcal{A}$ and PH be the set of paths for u. Tree u is deterministic if $\forall p \in PH \bullet |\{p' \in PH \mid trace(p) = trace(p')\}| = 1$.*

PPTs are not constrained to describe deterministic languages. This can be shown trivially with the valid PPT $\times(a\colon 1, a\colon 2)\colon 3$. Also note that this tree constructs a non-deterministic PPT from two DPPTs. Determinism is a desirable property in a model: it makes problems such as parsing and calculating the most probable path easier [152], and some conformance techniques are constrained to deterministic stochastic models [101, 14]. In this section we describe functions for calculating the determinism of a model.

As the operators $\rightarrow$ and $\circlearrowright_n$ are sequential in form, the only PPT operators which may introduce non-determinism are $\times$, $\wedge$ and $\circlearrowright_p$. We first introduce helper functions for identifying an alphabet and support for the empty trace.

Helper function abet is a notational convention which gives the alphabet for a PPT.

Helper function etr identifies whether a tree accepts an empty trace.

$$\begin{aligned}
&\text{Let etr}\colon \mathcal{PPT} \rightarrow \mathbb{B}\\
&\text{etr}(x\colon w) = (x = \tau) \text{ with } x \in A \cup \{\tau\}\\
&\text{etr}(\circlearrowright_p^{\rho}(u)) = \text{true}\\
&\text{etr}(\circlearrowright_n^{m}(u)) = \text{etr}(u)\\
&\text{etr}(\times(u_1,...,u_m)\colon w) = \text{etr}(u_1) \vee ... \vee \text{etr}(u_m)\\
&\text{etr}(\oplus(u_1,...,u_m)\colon w) = \text{etr}(u_1) \wedge ... \wedge \text{etr}(u_m) \text{ where } \oplus \in \{\rightarrow, \wedge\}
\end{aligned}$$

The α function identifies starting symbols.

$$\begin{aligned}
\alpha\colon \mathcal{PPT} &\to \mathbb{P}(A \cup \{\tau\}) \\
\alpha(x\colon w) &= \{x\} \text{ where } x \in A \cup \{\tau\} \\
\alpha(\to(u_1, ..., u_m)\colon w) &= \alpha(u_1) \cup \alpha(\to(u_2, ..., u_m)\colon w) \text{ if } \operatorname{etr}(u_1) \\
\alpha(\to(u_1, ..., u_m)\colon w) &= \alpha(u_1) \text{ if } \neg \operatorname{etr}(u_1) \\
\alpha(\oplus(u)) &= \alpha(u) \text{ where } \oplus \in \{\circlearrowleft_n, \circlearrowleft_p\} \\
\alpha(\times(u_1, ..., u_m)) &= \alpha(u_1) \cup ... \cup \alpha(u_m) \text{ where } \oplus \in \{\times, \wedge\}
\end{aligned}$$

The function det reports whether a tree is deterministic.

$$\begin{aligned}
\operatorname{det}\colon \mathcal{PPT} &\to \mathbb{B} \\
\operatorname{det}(x\colon w) &= \text{true where } x \in A \cup \{\tau\} \\
\operatorname{det}(\to(u_1, .., u_m)\colon w) &= \forall i \bullet \operatorname{det}(u_i) \\
&\quad \wedge \operatorname{det}(\to(u_2, .., u_m)\colon w) \\
&\quad \wedge \operatorname{etr}(u_1) \implies \alpha(u_1) \cap \alpha(\to(u_2, .., u_m)\colon w) = \varnothing \\
\operatorname{det}(\times(u_1, .., u_m)) &= \forall i \bullet \operatorname{det}(u_i) \\
&\quad \wedge \bigcup_{i \neq j} \alpha(u_i) \cap \alpha(u_j) = \varnothing \\
\operatorname{det}(\wedge(u_1, ..., u_m)\colon w) &= \forall i \bullet \operatorname{det}(u_i) \\
&\quad \wedge \bigcup_{i \neq j} \operatorname{abet}(u_i) \cap \operatorname{abet}(u_j) = \varnothing \\
\operatorname{det}(\oplus_1(u)) &= \operatorname{det}(u) \wedge \neg \operatorname{etr}(u) \\
&\quad \text{where } \oplus = \circlearrowleft_p \vee \oplus = \circlearrowleft_n^m \wedge m \geqslant 2 \\
\operatorname{det}(\circlearrowleft_n^1(u)\colon w) &= \operatorname{det}(u)
\end{aligned}$$

Subtrees which allow empty traces can make sequences and loops non-deterministic. As an example, consider the trace $\langle a, a \rangle$ and the PPT $\circlearrowleft_p^2 (\times(a\colon 2, \tau\colon 1))\colon 3$, where there are multiple paths through the silent activity producing the same overall trace.

The $\operatorname{det}(u)$ function visits each node at most once, so completes in $O(|u|)$ time.

4.4 Discovery - Toothpaste Miner

The *Toothpaste framework* describes reduction algorithms that "squeeze" an initial trace model into a more summarized and useful form using transformation rules. Starting with an event log, a trace model PPT is built, followed by the repeated application of transformation rules, until termination. The tree structure of a PPT, under certain rule guarantees, allows for a recursive, polynomial-time algorithm that can consider subtrees isolated from global impacts and have a straightforward termination condition. Section 4.4.1 and 4.4.2 give two instantiations, the Direct Toothpaste Miner (Definition 28) and the Incremental Toothpaste Miner (Definition 29). We organise and classify rules two ways: by information-preservation using quality criteria (in Section 4.4.3), and by impact on determinism (in Section 4.4.4).

4.4.1 Direct Toothpaste Miner

Toothpaste miner algorithms first transform an event log into an internal PPT. They then repeatedly transform the PPT by applying transformation rules. These rules reduce, summarize, or restructure the tree towards a desirable form. When desired criteria for an output process model are met, the PPT is translated into an SLPN as the final output. Criteria may include quality criteria such as fitness or precision thresholds, simplicity, or the preservation of certain critical trace paths in the final model. As the miner proceeds largely by reduction from an initial state which perfectly matches the event log, this allows for fine-grained control over what elements of the initial log are preserved in the final model. In the Direct Toothpaste Miner, a simple termination condition is used: that no more rules can be applied.

Event logs and the trace model

Given an event log over activities $A \subseteq \mathcal{A}$, a trace model PPT is given by $tm : \mathcal{L} \to \mathcal{PPT}$. Each trace is converted, with how often it occurs, by $st : A^* \times \mathbb{N} \to \mathcal{PPT}$.

$$\begin{aligned} st(\langle a_1, ..., a_m\rangle, j) &= \rightarrow(a_1\colon j, ..., a_m\colon j)\colon j \\ st(\langle a\rangle, j) &= a\colon j \\ tm([t_1^{i_1}, ..., t_m^{i_m}]) &= \times(st(t_1, i_1), ..., st(t_m, i_m))\colon |L| \text{ where } L = [t_1^{i_1}, ..., t_n^{i_n}] \end{aligned}$$

For example, $tm([\langle a, b\rangle, \langle c\rangle^3]) = \times(\rightarrow(a, b)\colon 1, c\colon 3)\colon 4$.

The trace model produced by function tm may produce a non-deterministic tree: for example, $tm([\langle a\rangle, \langle a, b\rangle]) = \times(a\colon 1, \rightarrow(a, b)\colon 1)\colon 2$. For non-trivial event logs, with many similar traces, it will be common for the trace model to be non-deterministic.

Transformation

In the Toothpaste framework, discovery consists of the application of transformation rules to a PPT, yielding progressively improved models. A rule is any function taking a PPT as input and producing another as output, that is, one with signature $r\colon \mathcal{PPT} \to \mathcal{PPT}$. We call those rules which reduce the number of nodes in a tree *reduction rules*. In this section, a basic Toothpaste miner is constructed by instantiating the framework using reduction rules.

Function *apply* applies a prioritised list of transformation rules to a PPT:

$$apply\colon \mathcal{PPT} \times (\mathcal{PPT} \to \mathcal{PPT})^* \to \mathcal{PPT}$$

$$\begin{aligned} apply(pt, \langle\rangle) &= pt \\ apply(pt, \langle r\rangle) &= r(pt) \\ apply(pt, \langle r\rangle + rs) &= apply(r(pt), rs)) \end{aligned}$$

Rules may not apply to every tree. If no rules in the rule list apply, the algorithm cannot further reduce the tree, and a local minimum has been reached. $apply_M$ finds a local reduction minimum by applying *apply* exhaustively:

$$apply_M : \mathcal{PPT} \times (\mathcal{PPT} \to \mathcal{PPT})^* \to \mathcal{PPT}$$

$$apply_M(pt, rs) = \begin{cases} apply_M(pt', rs) & \text{if } pt \neq pt' \\ pt & \text{otherwise} \end{cases}$$

$$\text{where } pt' = apply(pt, rs)$$

Both *apply* and $apply_M$ are guaranteed to terminate when used with reduction rules, as the size of the input tree is monotonically decreasing. Informally, so long as rules are chosen to preserve fidelity to the log, a minimal reducible model is desirable. The degree to which fidelity is desirable can be controlled by which rules are provided to the discovery algorithm. If a given ruleset is not confluent [27, p10], finding a globally minimal model is not guaranteed and can depend on the sequence in which the rules are applied.

These components are assembled into a process discovery algorithm.

Definition 28 (Direct Toothpaste miner)**.**

Given reduction rule sequence rs,

$$dtm \colon \mathcal{L} \times (\mathcal{PPT} \to \mathcal{PPT})^* \to \mathcal{SN}$$

$$dtm(L, rs) = apply_M(tm(L), rs)$$

is the Direct Toothpaste Miner.

That is, the Direct Toothpaste Miner exhaustively applies rules ($apply_M$) in sequence (rs) to a trace model ($tm(L)$) until terminating with an output PPT model.

Discovery example

An example of model discovery with *dtm* is in Figure 4.10, applying (and previewing) rules from Section 4.5. The trace model 4.10a has identical $\langle a, b\rangle$ traces consolidated in model 4.10b. The repeated c activities are summarized with a fixed loop in 4.10c. Concurrency of events a and b is identified in 4.10d. Finally, a probabilistic loop is introduced in 4.10e. Figure 4.11 shows the translation of the output PPT to an SLPN, a convenient form for other established process mining analysis techniques.

Complexity

The computational complexity of the *apply* algorithms depend on the size of the PPT data structure. Function *st* produces a tree with a depth of two, and $|t_i|+1$ nodes for an input trace t_i. The full trace model produced by *tm* adds a choice node, and in the worst case each trace is unique, for a total size (and memory complexity) bounded by $\Sigma(|t_i|+1)+1 = ||L|| + |L| + 1$ or $O(||L||)$.

The complexity of *apply* depends on evaluating each node of the tree with reduction rules. If each sub-tree can be summarized with one traversal, the worst case is comparing each node to

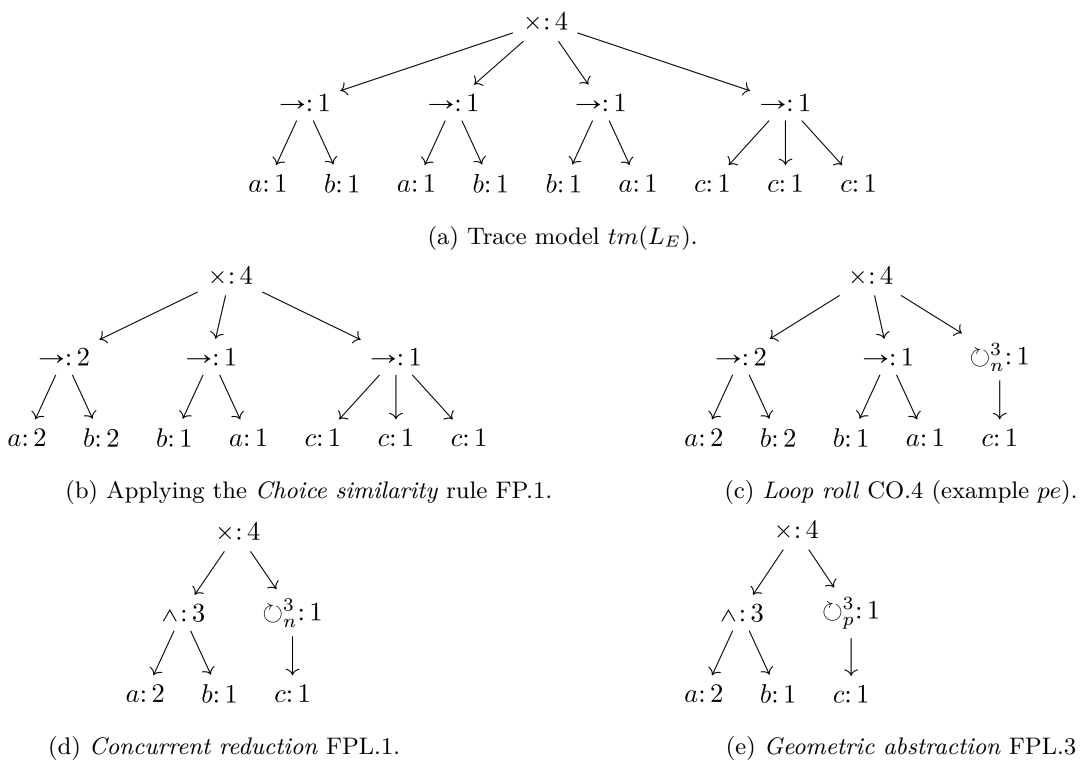


Figure 4.10: Discovery example using the Direct Toothpaste Miner (*dtm*).

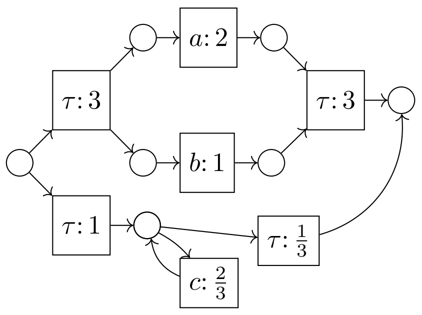


Figure 4.11: *SLPN output* for *dtm*(L_E), seen in Figure 4.10.

each other node, giving $O(apply(u, R)) = O(|u|^2|R|)$ comparisons. Writing u_L for the full PPT trace model produced by tm, this is limited by $(2||L||)^2|R|$ or $O(apply(u_L, R)) = O(||L||^2|R|)$.

Applying *apply* exhaustively with $apply_M$ requires executing this process a number of times. So long as the rule list R consists of solely reduction rules, then the size of the tree is monotonically decreasing. The worst case for time complexity is then also the best case for model size reduction and is bounded by the size of the trace model, $2||L||$. This yields $O(apply_M(L, R)) = O(||L||^3|R|)$. The overall worst-case time complexity is then dominated by the cubic term and is $O(||L||^3|R|)$.

4.4.2 Incremental Discovery

Incremental process discovery is the construction of a model from a stream of events, rather than a static event log fully known before invocation of the discovery algorithm. An incremental Toothpaste Miner is introduced below.

Definition 29 (Incremental Toothpaste miner)**.** *Given trace $\sigma \in A^*$, rule list rs, existing model $u_M \in \mathcal{PPT}$, and function st per Definition 28, incremental miner $apply_\Delta$ is:*

$$apply_\Delta \colon \mathcal{PPT} \times A^* \times \mathbb{N} \times (\mathcal{PPT} \to \mathcal{PPT})^* \to \mathcal{PPT}$$
$$apply_\Delta(x\colon w, \sigma, j, rs) = apply_M(\times(x\colon w, st(\sigma, j))\colon w + j, rs)$$

An entire event log L may be presented as a stream.

Definition 30 (Repeated Incremental Toothpaste Miner)**.**

$$\begin{aligned} di &\colon \mathcal{L} \times (\mathcal{PPT} \to \mathcal{PPT})^* \to \mathcal{PPT} \\ di(L, rs) &= apply_\Delta(apply_\Delta(L - \{\sigma \mapsto j\}, rs), \sigma, j, rs), \textit{ for some } \sigma^j \in L \\ di([\sigma^j], rs) &= apply_M(st(\sigma, j), rs) \end{aligned}$$

As for other Φ and *dtm* algorithms, the time complexity of *di* is dependent on the size of the tree. For the first trace t_1, the size of the initial model is $|t_1| + 1$. In the worst case, the size of the process model increases over time, so that subsequent traces t_i add $|t_i|$ nodes each. The time for each run of $apply_\Delta$ is $|t_i|^2|R|$. The overall worse case size for a log L is then

$$\begin{aligned} O_{mem}(di(L, rs)) &= \Sigma_{i=1}^{|L|}|t_i||R| = |R|\Sigma_{i=1}^{|L|}|t_i| \\ &= O(|R| \cdot ||L||), \text{ by the definition of} ||L|| \end{aligned}$$

An upper bound for time complexity can be found using Lemma 1, a simple result which we somewhat pedantically re-prove for lack of a reference.

Lemma 1. *For $N \subseteq \mathbb{N}$, $\sum\limits_{n \in N} n^2 \leqslant (\sum\limits_{n \in N} n)^2$.*

Proof. By induction over $|N|$. The $|N| = 0$ case holds trivially. Initial case $|N| = 1$, for which

$a^2 = (a)^2$,

$$\text{Show} \sum_{n\in N} n^2 \leqslant (\sum_{n\in N} n)^2 \implies \sum_{n\in N\cup\{m\}} n^2 \leqslant (\sum_{n\in N\cup\{m\}} n)^2$$

$$(\sum_{n\in N\cup\{m\}} n)^2 = (\sum_{n\in N} n + m)^2 = (\sum_{n\in N} m)^2 + 2m(\sum_{n\in N} n) + m^2$$

$$\sum_{n\in N} n^2 \leqslant (\sum_{n\in N} n)^2 \implies \sum_{n\in N} n^2 + m^2 \leqslant (\sum_{n\in N} n)^2 + m^2 + 2m(\sum_{n\in N} n)$$

$\square$

Then, $O_{time}(di(L,rs)) = \Sigma_{i=1}^{|L|} 2|t_i|^2|R| = 2|R|\Sigma_{i=1}^{|L|}|t_i|^2 \leqslant 2|R|(\Sigma_{i=1}^{|L|}|t_i|)^2$ by Lemma 1 $\leqslant 2|R|||L||^2$, by definition of $||L||$. So $O_{time}(di(L,rs)) \leqslant O(||L||^2|R|)$.

Notably, the time complexity of the incremental algorithm *di* is quadratic, rather than the cubic complexity of reducing the entire trace model at once with *dtm*. Informally, the smaller size of the model at the start of the execution of the algorithm results in time savings compounding. An important design trade-off remains, as stochastic information loss occurs with most reduction rules, and some classes of rules cause more information loss than others.

4.4.3 Classification By Quality Criteria

Transformation rules are classified by their impact on standard process model quality criteria of fitness, precision, and simplicity [6, p189], and by the criteria of stochastic information loss. Standard process model quality criteria are control-flow criteria and do not take the stochastic perspective into account. Reduction rules are categorized in four cuts: Preserving Compression, Fitness- and Precision-Preserving, Fitness-Preserving, and Simplifying Lossy, as seen in Figure 4.12. The categories and their inter-dependencies are detailed below. These are general categories across all rules, which will be applied to classify specific rules in Section 4.5.

Consider log L with language TL_L and stochastic language Θ_L, and model M with language TL_M and stochastic language Θ_M. Fitness is given by $ft(TL_M, TL_L) = \frac{|TL_M \cap TL_L|}{|TL_L|}$.

In defining precision, we have to account for infinitely many traces, due to the $\circlearrowleft_p$ construct [144]. Low-probability traces which are longer than the number of events in the log are filtered out in truncated language TL_{TM}. We also restrict the definition to trees with k-bounded silence, where $k = ||L||$. This allows imprecisely defined nets to still be punished for misallocation of probability mass to long traces not found in the log, without allowing infinite silent paths.

$$TL_{TM} = \{\sigma \in TL_M \mid |\sigma| < ||L|| \vee \Theta_M(t) > \epsilon\} \text{ where } 0 < \epsilon \ll 1$$

Precision is then given by $pn(TL_M, TL_L) = \frac{|TL_{TM} \cap TL_L|}{|TL_{TM}|}$.

As a categorization tool for process model transformation rules, fitness and precision are helpful in showing the loss or retention of information, even if they are insensitive to stochastic information. The classification of reduction rules is of particular interest, and necessary to maintain the monotonically simplifying property of the Toothpaste algorithm in Section 4.4.1.

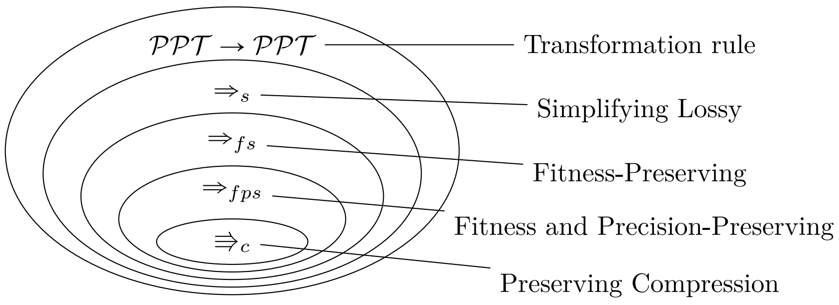


Figure 4.12: Rule categories by information preservation.

No Loss Of Fitness Or Precision Without Loss of Stochastic Information. There are no categories of "Fitness- and Stochastic Information-Preserving But Precision-Reducing" or "Precision- and Stochastic Information-Preserving but Fitness Reducing" transformation rules. We say there is *stochastic fidelity* between stochastic languages Θ_1, Θ_2 when $\forall \sigma \bullet \Theta_1[\sigma] = \Theta_2[\sigma]$. Models that no longer have stochastic fidelity have experienced stochastic information loss.

It is not possible to maintain fitness and reduce precision without also losing stochastic information from a model.

Theorem 1. *Given log L, models M and M', and corresponding stochastic languages $\Theta_M, \Theta_{M'}$:*

$$ft(M,L) = ft(M',L) \wedge pn(M,L) > pn(M',L) \implies \exists \sigma \bullet \Theta_M[\sigma] \neq \Theta_{M'}[\sigma]$$

Proof. Take log $L \in \mathcal{L}$ with trace language TL_L and stochastic language Θ_L, and process model M, with trace language TL_M and stochastic language Θ_M. Let M' be a second model covering language $TL_{M'}$ such that $ft(M,L) = ft(M',L) \wedge pn(M,L) > pn(M',L)$. By the fitness definition, $\frac{|TL_M \cap TL_L|}{|TL_L|} = \frac{|TL_{M'} \cap TL_L|}{|TL_L|}$. As precision decreases, there must therefore be at least one new trace $\sigma_1 \in M' \wedge \sigma_1 \notin M$, which is equivalent to $\Theta_M(\sigma_1) = 0$ and $\Theta_{M'}(\sigma_1) > 0$. As probabilities must sum to one, some other trace $\sigma_2 \in M$ must have reduced in probability. $\exists \sigma_2 \in TL_M \bullet \Theta_{M'}(\sigma_2) < \Theta_M(\sigma_2)$.

Case 1: Stochastic fidelity had been retained. $\Theta_M[\sigma_2] = \Theta_L[\sigma_2] \neq \Theta_{M'}[\sigma_2]$.

Case 2: Stochastic fidelity already lost. $\Theta_M[\sigma_2] \neq \Theta_L[\sigma_2]$. The trace σ_1 holds no information for restoring stochastic information on σ_2, as σ_2 is not an element of the log L or covered by the original model M. □

Transformation rules only have information from their input model parameter. There is no further log or trace parameter to the rules, making systematic restoration of stochastic fidelity impossible once it has been lost.

If $\frac{|TL_{M'}|}{|TL_M|} < \frac{|TL_L \cap TL_{M'}|}{|TL_L \cap TL_M|}$, and a rule reduces fitness, then precision increases. This would make a sub-category of Simplifying Lossy Rules; however, no useful concrete rules were found in this sub-category.

Trace Language Expansion Preserves Fitness Fitness is preserved when transforming M to M' when the trace languages have the relation $\mathbf{tg}(M) \subseteq \mathbf{tg}(M')$. This can be seen for log $L \in \mathcal{L}$ with language λ_L as $\frac{|\mathbf{tg}(M) \cap \lambda_L|}{|\lambda_L|} \leqslant \frac{|\mathbf{tg}(M') \cap \lambda_L|}{|\lambda_L|}$.

Table 4.2: PPT transformation rule classification by impact on determinism, given $u' = tr(u)$ for some rule tr.

Classification	Definition	Description
α-reducing	$\alpha(u') \subseteq \alpha(u) \wedge$ $\alpha_{nd}(u') \subseteq \alpha_{nd}(u)$	Separates and restricts starting activities
α-stable	$\alpha(u') = \alpha(u)$	No change to starting activities
determinism-trap	$\det(u) \implies \det(u')$	Never introduces non-determinism. Superset of preceding types.

Precision is trivially preserved when $\mathbf{tg}(M) = \mathbf{tg}(M')$, with other transformations requiring specific examination by cases.

4.4.4 Rule Determinism

Since Probabilistic Process Trees (PPTs) have a finite automata semantics, Deterministic Probabilistic Process Trees (DPPTs) are Stochastic Deterministic Finite Automata (SDFAs). SDFAs are not closed under union [152] and DPPTs are not closed under $\times$; trivially, $\times(a\colon 1, a\colon 2)\colon 3$ combines two deterministic sub-trees. The determinism functions α and det are used to classify rules by their impact on determinism in Table 4.2, and described below.

To help in rule classification, the α_{nd} function identifies non-determinant symbols across the entire tree.

$$\alpha_{nd}\colon \mathcal{PPT} \to \mathbb{P}(\mathcal{A} \cup \{\tau\})$$
$$\alpha_{nd}(x\colon w) = \varnothing \text{ where } a \in A \cup \{\tau\}$$
$$\alpha_{nd}(\oplus(u_1, ..., u_n)\colon w) = \alpha(u_1) \cap ... \cap \alpha(u_n) \text{ where } \oplus \in \{\times, \wedge\}$$
$$\alpha_{nd}(\to(u_1, ..., u_n)\colon w) = \alpha_{nd}(u_1) \cup ... \cup \alpha_{nd}(u_n)$$
$$\alpha_{nd}(\oplus_1(u)) = \alpha_{nd}(u) \text{ where } \oplus \in \{\circlearrowleft_p, \circlearrowleft_n\}$$

Non-determinant symbols identified by α_{nd} are those "exposed" as starting symbols outside the block, potentially causing non-determinism.

Rules which maintain DPPT closure are termed *determinism-trap*s.

Definition 31 (Determinism-trap)**.** *A determinism-trap is a transformation rule which preserves determinism:* $\forall u \in \mathcal{PPT}, tr\colon (\mathcal{PPT} \to \mathcal{PPT})^* \bullet tr$ *is a determinism-trap* $\iff \det(u) \implies \det(tr(u))$.

Maintaining determinism with a transformation rule depends on the α function for possible starting symbols, not just the determinism of the subtree given by det. Theorems 2 and 3 explore this.

Theorem 2 (α-stable rules preserve determinism)**.** *Let* $tr \in \mathcal{PPT} \to \mathcal{PPT}$ *be an* α*-stablea transformation rule.* $\forall u \in \mathcal{PPT} \bullet \det(u) \implies \det(tr(u))$.

That is, application of α-stable rules to a sub-tree preserves the determinism of the parent tree.

Proof. For α-stable rules tr, for PPTs $\oplus(u_1, ..., u_n)$, as $\forall i \bullet \alpha(tr(u_i)) \subseteq \alpha(u_i)$, the intersection $\alpha(u_i) \cap \alpha(u_j)$ does not increase. $\det(u)$ for $\times$ depends on this pairwise intersection being empty, and $\wedge$ has a stricter constraint using intersections of the child alphabets (via abet). For sequences, the starting symbols for subsequence $\rightarrow(u_2, ..., u_m)$ cannot increase without the pairwise intersections increasing. The leaf and loop cases are trivial. Therefore $\det(u) \implies \det(tr(u))$, the definition of a determinism-trap. □

Corollary 1 (α-reducing rules preserve determinism)**.** *Let* $tr \in \mathcal{PPT} \rightarrow \mathcal{PPT}$ *be an* α*-reducing a transformation rule.* $\forall u \in \mathcal{PPT} \bullet \det(u) \implies \det(tr(u))$.

Proof. By the same reasoning as Theorem 3. □

A transformation rule can preserve determinism on all PPTs, but when applied to a subtree, can cause non-determinism in the enclosing tree.

Theorem 3 (Determinism-trap rules may cause non-determinism in parents)**.** $\exists tr \in \mathcal{PPT} \rightarrow \mathcal{PPT}$ *such that* tr *is a determinism-trap, and*

$$\exists u_{sub}, u, u' \in \mathcal{PPT} \bullet u_{sub} \hat{\subseteq} u \wedge tr(u) \hat{\subseteq} u' \wedge \det(u) \wedge \neg \det(u')$$

Proof. Consider the possible transformation rule $nb(\rightarrow(a\colon 2, b\colon 2)\colon 2) = \times(a\colon 1, b\colon 1)\colon 2$. (Note this is a simple transformation used for this proof, and not a rule used for our discovery algorithm.) The rule preserves determinism, as $\det(nb(x)) \wedge \det(x)$. When applied to the subtree of $\times(b\colon 1, \rightarrow(a\colon 2, b\colon 2)\colon 2)\colon 3$, the new tree is non-deterministic. □

Applying these results does not guarantee the discovery of a deterministic model, but they can maintain the determinism of a reduced model once one is discovered.

Trace Language Preservation Implies Alpha-Stability When rules preserve the trace language of a model, they are α-stable.

Lemma 2. $\forall tr \in \mathcal{PPT} \rightarrow \mathcal{PPT} \bullet \mathbf{tg}(u) = \mathbf{tg}(tr(u)) \implies$ *tr is* α*-stable.*

Proof. Consider transformation rule tr where $\forall u \in \mathcal{PPT} \bullet \mathbf{tg}(u) = \mathbf{tg}(tr(u))$. Assume tr is not α-stable. Then, by the definition of α-stable, there is some set of activities and some PPT that invalidates α-stability.

$$\exists A \subseteq \mathcal{A}, u \in \mathcal{PPT}, a \in A \bullet a \in \alpha(u) \cup \alpha(tr(u)) \wedge a \notin \alpha(u) \cap \alpha(tr(u))$$

Given such an activity a, it must be at the head of some trace.

$$\exists \sigma \in A^* \bullet \sigma = \langle a, ... \rangle \wedge \sigma \notin \mathbf{tg}(u) \cap \mathbf{tg}(tr(u))$$

That is, this σ trace is not in the trace language of the original but is in the trace language of the transformed model. This contradicts the initial assumption of tr, so preserving the trace language entails α-stability. □

4.5 Concrete Rules

This section introduces concrete transformation rules, arranged by information-preservation category. Each rule listing includes a formal statement of the rule, the determinism category, proof of the rule operation and categorisation, and an example of its use. These concrete rules have wide coverage, but do not exhaust all possible rules. All of the concrete rules presented here, and used in the Toothpaste Miner, maintain the weight at the root node of the transformed PPT. This means that even when the stochastic language is not preserved, the new subtree has approximately the same probabilistic mass. Lastly we define a normal form for PPTs using the rules.

In each subsection, we introduce the information-preservation category, then each rule in the category. The impact on determinism, using the system in Table 4.2, is also shown.

To investigate the properties of these rules, we define a stochastic preservation relation, $\overset{\sim}{\Longleftrightarrow}$.

Definition 32 (Stochastic Preservation Relation)**.** *Let PTR be a set of transformation rules that preserve stochastic languages, and* **slang** *a function that gives the stochastic language for a PPT. Let* $tr(u_2 \mid u_1)$ *denote the result of applying rule* tr *to subtree* u_2 *in PPT* u_1*, where* $u_2 \hat{\subseteq} u_1$*.*

$$\begin{aligned} PTR = \{tr \in & \mathcal{PPT} \rightarrow \mathcal{PPT} \mid \\ & \forall u_1 \in \mathcal{PPT} \bullet \mathbf{slang}(u_1) = \mathbf{slang}(tr(u_1)) \\ & \wedge \forall u_2 \hat{\subseteq} u_1 \bullet \mathbf{slang}(tr(u_2 \mid u_1)) = \mathbf{slang}(u_1)\} \end{aligned}$$

PTR is the set of rules which preserve stochastic languages when applied to any PPT, including when that PPT is a subtree of another.

Then $\overset{\sim}{\Longleftrightarrow}$*, or* stochastic preservation*, is the equivalence relation on PPTs such that*

$$\forall u_1, u_2 \in \mathcal{PPT} \bullet u_1 \overset{\sim}{\Longleftrightarrow} u_2 \equiv \exists tr \in PTR \bullet tr(u_1) = u_2 \vee tr(u_2) = u_1$$

As well as stochastic preservation being a useful property, $\overset{\sim}{\Longleftrightarrow}$ lets us show the properties of rules in an algebraic style, by making statements such as $\times(a\colon 1, b\colon 2)\colon 3 \overset{\sim}{\Longleftrightarrow} \times(b\colon 2, a\colon 1)\colon 3$.

To aid in defining these rules, we next introduce helper functions.

4.5.1 Helper Functions

These functions and relations help define transformation rules. All functions preserve the trace language, so are α-stable by Lemma 2.

A scaling function, $\Gamma(u, \gamma)$ multiplies every weight in the tree by γ.

$$\begin{aligned} \Gamma\colon \mathcal{PPT} &\rightarrow \mathcal{PPT} \\ \Gamma(x\colon w, \gamma) &= x\colon \gamma w \text{ where } x \in \mathcal{A} \cup \{\tau\} \\ \Gamma(\oplus(u_1, ..., u_m)\colon w, \gamma) &= \oplus(\Gamma(u_1, \gamma), ..., \Gamma(u_m, \gamma))\colon \gamma w \end{aligned}$$

Γ makes no changes to symbols, so preserves α-stability.

PPTs are *similar*, denoted $\cong_c$, when only weights need to be changed to make them strictly equal. This also implies they share the same trace language.

$$\begin{aligned}
&\cong_c : \mathcal{PPT} \leftrightarrow \mathcal{PPT}\\
&x\colon w_1 \cong_c x\colon w_2, \text{ where } x \in \mathcal{A} \cup \{\tau\}\\
\oplus_1(u_{x1}, ..., u_{xn})\colon w_1 \cong_c{}& \oplus_2 (u_{y1}, ..., u_{yn})) : w_2\\
&\Leftrightarrow \oplus_1 = \oplus_2 \wedge \forall i \in [1, .., n] \bullet u_{xi} \cong_c u_{yi}
\end{aligned}$$

Similarity is both associative and commutative. As an example, $\times(a\colon 1, b\colon 2)\colon 3 \cong_c \times(a\colon 5, b\colon 8)\colon 13$.

A stochastic merge function, Ψ, combines two similar trees by adding weights, with $\circlearrowright_p$ repetitions being scaled by relative weight. Ψ preserves control-flow fitness and precision of the input process trees, but loses stochastic information, unless $x = y$ for $\Psi(x, y)$.

$$\begin{aligned}
&\Psi : \mathcal{PPT} \times \mathcal{PPT} \to \mathcal{PPT}\\
&\Psi(x\colon w_1, x\colon w_2) = x\colon w_1 + w_2, \text{ where } x \in \mathcal{A} \cup \{\tau\}\\
\Psi(\oplus(u_{x1}, .., u_{xn}) : w_1, \oplus(u_{y1}, .., u_{yn}) : w_2) =& \oplus (\Psi(u_{x1}, u_{y1}), ..., \Psi(u_{xn}, u_{yn}))\colon w_1 + w_2\\
&\text{where } \forall i \in [1, .., n] \bullet u_{xi} \cong_c u_{yi}\\
\Psi(\circlearrowright_p^{\rho_1} (x_1\colon w_1), \circlearrowright_p^{\rho_2} (x_2\colon w_2)) =& \circlearrowright_p^{\rho'} (\Psi(x_1\colon w_1, x_2\colon w_2))\\
&\text{where } \rho' = \frac{\rho_1 w_1 + \rho_2 w_2}{w_1 + w_2}
\end{aligned}$$

As an example, $\Psi(\to(a, b)\colon 2, \to(a, b)\colon 3) = \to(a, b)\colon 5$.

4.5.2 Preserving Compressions

The following rules are information-preserving reduction rules, achieving compression by using a smaller tree to describe the same stochastic language. They are denoted with $\Rrightarrow_c$ and are all α-stable by Lemma 2.

• **Single node collapse** Choice and sequence nodes with only one child node can always be collapsed to that child node.

$$\oplus(x\colon w) : w \Rrightarrow_c x\colon w \text{ where } \oplus \in \{\to, \times\} \tag{CO.1}$$

This holds as the stochastic languages for these operators are preserved across single nodes. $\to(u) \overset{\sim}{\Longleftrightarrow} \times(u) \overset{\sim}{\Longleftrightarrow} u$.

An example use:
$$\begin{tikzcd}
\to\colon 1 \arrow[d] & \Rrightarrow_c & a\colon 1\\
a\colon 1 & &
\end{tikzcd}$$

• **Flatten** Nodes with a multi-child operator, and children which are also trees with the same operator, can be brought under a single parent node.

$$\oplus(u_1, \oplus(u_2, u_3))\colon w \Rrightarrow_c \oplus(u_1, u_2, u_3)\colon w \text{ where } \oplus \in \{\to, \times\} \tag{CO.2}$$

This holds as a straightforward consequence of operator associativity.

An example use:
$$\begin{tikzcd}[column sep=small] & \times\colon 6 \arrow[dl] \arrow[dr] & & & & \times\colon 6 \arrow[dl] \arrow[d] \arrow[dr] & \\ a\colon 1 & & \times\colon 5 \arrow[dl] \arrow[dr] & \Rrightarrow_c & a\colon 1 & b\colon 2 & c\colon 3 \\ & b\colon 2 & & c\colon 3 & & & \end{tikzcd}$$

- **Fixed loop identity** A single-iteration fixed loop is equivalent to its subtree.

$$\circlearrowright_n^1(u) : w \Rrightarrow_c u\colon w \tag{CO.3}$$

Stochastic languages for single iteration loops are the same as those for the child node, or a sequence with one child:

$$\begin{aligned} &\circlearrowright_n^1(u)\colon w \overset{\sim}{\Longleftrightarrow} \rightarrow(u) \overset{\sim}{\Longleftrightarrow} u \\ &\circlearrowright_n^1(u) : w \Rrightarrow_c u\colon w \end{aligned}$$

An example use:
$$\begin{tikzcd} \circlearrowright_n^1\colon 3 \arrow[d] & \Rrightarrow_c & a\colon 3 \\ a\colon 3 & & \end{tikzcd}$$

- **Fixed loop roll** Sequences of the same subtree can be summarized as fixed loops.

$$\begin{aligned} \rightarrow(u, u)\colon w &\Rrightarrow_c \circlearrowright_n^2(u)\colon w \\ \rightarrow(u, \circlearrowright_n^m(u))\colon w &\Rrightarrow_c \circlearrowright_n^{m+1}(u)\colon w \\ \rightarrow(\circlearrowright_n^m(u), u)\colon w &\Rrightarrow_c \circlearrowright_n^{m+1}(u)\colon w \\ \rightarrow(\circlearrowright_n^{m_1}(u), \circlearrowright_n^{m_2}(u))\colon w &\Rrightarrow_c \circlearrowright_n^{m_1+m_2}(u)\colon w \end{aligned} \tag{CO.4}$$

Note that $\circlearrowright_n^m(u)\colon w \overset{\sim}{\Longleftrightarrow} \rightarrow(u ... m \text{ times } ...)\colon w$ by the definition of the fixed loop operator $\circlearrowright_n$ in Definition 25. So this rule achieves compression without information loss.

An example use:
$$\begin{tikzcd}[column sep=small] & \rightarrow\colon 2 \arrow[dl] \arrow[dr] & & \Rrightarrow_c & \circlearrowright_n^4\colon 2 \arrow[d] \\ a\colon 2 & & \circlearrowright_n^3\colon 2 \arrow[d] & & a\colon 2 \\ & & a\colon 2 & & \end{tikzcd}$$

- **Fixed loop nesting** Nested fixed loops can be consolidated.

$$\circlearrowright_n^{m_1}(\circlearrowright_n^{m_2}(u))\colon w \Rrightarrow_c \circlearrowright_n^{m_1 m_2}(u)\colon w \tag{CO.5}$$

Stochastic language preservation follows directly from the definition of fixed loops:

$$\begin{aligned} \circlearrowright_n^{m_1}(\circlearrowright_n^{m_2}(u))\colon w &\equiv \rightarrow(\underbrace{\rightarrow(u, \dots\ m_2 \text{ times } \dots), \dots \rightarrow(u, \dots\ m_2 \text{ times } \dots)\dots}_{m_1 \text{ times}}) \\ &\overset{\sim}{\Longleftrightarrow} \circlearrowright_n^{m_1 m_2}(u)\colon w \end{aligned}$$

An example use:
$$\begin{tikzcd} \circlearrowright^{5}_{n}\colon 1 \arrow[d] \arrow[r, phantom, "\Rrightarrow_c"] & \circlearrowright^{15}_{n}\colon 1 \arrow[d] \\ \circlearrowright^{3}_{n}\colon 1 \arrow[d] & b\colon 1 \\ b\colon 1 & \end{tikzcd}$$

4.5.3 Fitness and Precision-Preserving With Stochastic Information Loss

For these rules, stochastic information is preserved only where sub-trees are strictly equal, as for Ψ. They are denoted with $\Rightarrow_{fps}$. All rules are α-stable by Lemma 2; some are also α-reducing.

Lemma 3 (Sequence Distribution Over Choice Preserves Trace Language)**.**

$$\rightarrow(u_1, \times(u_2, u_3)\colon w_2 + w_3)\colon w_2 + w_3 \cong_c \times(\rightarrow(u_1, u_2)\colon w_2, \rightarrow(u_1, u_3)\colon w_3)\colon w_2 + w_3$$

Proof. Per Definition 25, a sequence $\rightarrow$ enforces a strict trace order, and choice $\times$ is disjoint. Note also that the underlying automata $E_1 + \!\!\!+ (E_2 \sqcup E_3)$ and $(E_1 + \!\!\!+ E_2) \sqcup (E_1 + \!\!\!+ E_3)$ are disjoint after a new initial place where every exiting transition is an initial transition in E_1. □

• **Choice similarity reduction** Choices between structurally similar trees are merged into a single tree.

$$\times(u_1, u_2) \Rightarrow_{fps} \Psi(u_1, u_2) \text{ where } u_1 \cong_c u_2 \tag{FP.1}$$

This rule is α-stable.

Similar PPTs are trace language-equivalent, but not stochastic language equivalent, as similarity ($\cong_c$) compares each leaf and operator node, but not their weights. This shows α-stability. The fitness and precision of similar PPTs will then also be equal for the same log. For the same reason, the Ψ (merge) operator also only affects stochastic information, not fitness and precision.

Recall $\mathbf{wa}(\times(u_1, u_2)) = \mathbf{wa}(u_1) \sqcup \mathbf{wa}(u_2)$. No new traces are introduced or removed by $\sqcup$, and $u_1 \cong_c u_2$, so fitness and precision are preserved.

An example use:
$$\begin{tikzcd} & \times\colon 3 \arrow[dl] \arrow[dr] \arrow[r, phantom, "\Rightarrow_{fps}"] & & a\colon 3 \\ a\colon 1 & & a\colon 2 & \end{tikzcd}$$

• **Choice folding** Pulls up common prefixes in a choice into the head of a sequence.

$$\begin{aligned} &\times(\rightarrow(u_{x1}, u_2)\colon w_1, \rightarrow(u_{x2}, u_3)\colon w_2)\colon w_1 + w_2 \\ &\Rightarrow_{fps} \rightarrow(\Psi(u_{x1}, u_{x2}), \times(u_2, u_3))\colon w_1 + w_2 \qquad \text{where } u_{x1} \cong_c u_{x2} \end{aligned} \tag{FP.2}$$

The correctness of this rule follows from:

$$\text{Take } \times(\rightarrow(u_{x1}, u_2)\colon w_1, \rightarrow(u_{x2}, u_3)\colon w_2)\colon w_1 + w_2))$$

Substitute $u_m = \Psi(u_{x1}, u_{x2})$ with loss of stochastic information

$$\begin{aligned} &\Rightarrow_{fps} \times(\rightarrow(u_m, u_2)\colon w_1, \rightarrow(u_m, u_3)\colon w_2)\colon w_1 + w_2 \\ &\cong_c \rightarrow(u_m, \times(u_2, u_3))\colon w_1 + w_2 \qquad \text{by Lemma 3} \end{aligned}$$

As the trace language of the tree is unchanged by this rule, fitness and precision are preserved. The original choice PPT is non-deterministic, but the result introduces no new symbols to α, and is deterministic when $\det(u_m) \wedge \det(\times(u_2, u_3)\colon w_1 + w_2)$. Hence, the rule is α-reducing.

An example use:

$$\begin{tikzcd}
 & & & \times{:}\,2 \arrow[dll] \arrow[drr] & & & & \Rightarrow_{fps} & & \to{:}\,2 \arrow[dl] \arrow[dr] & & \\
 & \to{:}\,1 \arrow[dl] \arrow[dr] & & & & \to{:}\,1 \arrow[dl] \arrow[dr] & & & a{:}\,2 & & \times{:}\,2 \arrow[dl] \arrow[dr] & \\
a{:}\,1 & & b{:}\,1 & & a{:}\,1 & & c{:}\,1 & & & b{:}\,1 & & c{:}\,1
\end{tikzcd}$$

• **Choice folding suffixes** Common suffixes are folded into a new sequence node at the tail.

$$\begin{aligned}
&\times((\to(u_1, u_{y1})\colon w_1), (\to(u_2, u_{y2})\colon w_2))\colon w_1 + w_2 \\
\Rightarrow_{fps} &\to(\times(u_1, u_2), \Psi(u_{y1}, u_{y2}))\colon w_1 + w_2 \text{ where } u_{y1} \cong_c u_{y2}
\end{aligned} \tag{FP.3}$$

The correctness of this rule follows from:

$$\begin{aligned}
\text{Take } &\times((\to(u_1, u_{y1})\colon w_1), (\to(u_2, u_{y2})\colon w_2)) : w_1 + w_2 \\
\text{Substitute } u_m = \Psi(u_{y1}, u_{y2}) &\text{ with loss of stochastic information} \\
\Rightarrow_{fps} &\times((\to(u_1, u_m)\colon w_1), (\to(u_2, u_m)\colon w_2)) : w_1 + w_2 \\
\cong_c &\to(\times(u_1, u_2), u_m)\colon w_1 + w_2 && \text{by Lemma 3}
\end{aligned}$$

As the trace language of the tree is unchanged by this rule, fitness and precision are preserved. Starting symbols are unchanged, so the rule is α-stable.

An example use:

$$\begin{tikzcd}
 & & & \times{:}\,3 \arrow[dll] \arrow[drr] & & & & \Rightarrow_{fps} & & \to{:}\,3 \arrow[dl] \arrow[dr] & \\
 & \to{:}\,2 \arrow[dl] \arrow[dr] & & & & \to{:}\,1 \arrow[dl] \arrow[dr] & & & \times{:}\,3 \arrow[dl] \arrow[dr] & & b{:}\,3 \\
a{:}\,2 & & b{:}\,2 & & c{:}\,1 & & b{:}\,1 & a{:}\,2 & & c{:}\,1 &
\end{tikzcd}$$

• **Choice skip** A common head is pulled into a sequence with a choice between the tail and a silent activity.

$$\begin{aligned}
&\times(x_1\colon w_1, \to(x_2, y_1, ...)\colon w_2, \to(x_3, y_2, ...)\colon w_3) : v \\
\Rightarrow_{fps} &\to(\Psi(x_1\colon w_1, x_2\colon w_2, x_3\colon w_3), \times(y_1\colon w_2, y_2\colon w_3 ..., \tau\colon w_1))\colon v \\
&\text{where } x_1\colon w_1 \cong_c x_2\colon w_2 \cong_c x_3\colon w_3
\end{aligned} \tag{FP.4}$$

Note that applying this rule to two similar subtrees does not always result in a reduction of nodes. The rule is only applied when it would result in a reduction of the number of nodes. The quality criteria properties of the rule can be seen as follows.

$$
\begin{aligned}
&\text{Take } \times(x_1\colon w_1, \to(x_2, y_1, ...)\colon w_2, \to(x_3, y_2, ...)\colon w_3)\colon v\\
&\quad\text{Substitute } u_m = \Psi(x_1\colon w_1, x_2\colon w_2, x_3\colon w_3) \text{ for } x_1, x_2 \text{ and } x_3\\
&\quad\quad \Rightarrow_{fps}\ \times(u_m, \to(u_m, y_1, ...)\colon w_2, \to(u_m, y_2, ...)\colon w_3)\colon v\\
&\quad\text{Introduce silent transition suffix for solo } u_m\text{, which maintains trace language}\\
&\quad\quad \cong_c \times(\to(u_m, \tau)\colon w_1, \to(u_m, y_1, ...)\colon w_2, \to(u_m, y_2, ...)\colon w_3)\colon v\\
&\quad\quad\text{Apply Choice folding FP.2}\\
&\quad\quad \Rightarrow_{fps} \to(u_m, \times(y_1\colon w_2, y_2\colon w_3 ..., \tau\colon w_1))\colon v
\end{aligned}
$$

As the trace language of the tree is unchanged by this rule, fitness and precision are preserved. The original choice PPT is non-deterministic, but the result introduces no new symbols to α, and is deterministic when $\det(u_m) \wedge \det(\times(y_1\colon w_2, y_2\colon w_3 ..., \tau\colon w_1)\colon v)$. Hence the rule is α-stable. A corresponding rule for sequence suffixes exists, but is not a determinism-trap, so is not included.

An example use:

$$\begin{tikzcd}
 & & & \times\colon 5 \arrow[dlll] \arrow[dl] \arrow[drr] & & & & \Rightarrow_{fps} & & \to\colon 5 \arrow[dl] \arrow[dr] & & \\
a\colon 1 & & \to\colon 2 \arrow[dl] \arrow[dr] & & & \to\colon 2 \arrow[dl] \arrow[dr] & & & a\colon 5 & & \times\colon 5 \arrow[dl] \arrow[d] \arrow[dr] & \\
 & a\colon 2 & & b\colon 2 & a\colon 2 & & c\colon 2 & & & b\colon 2 & c\colon 2 & \tau\colon 1
\end{tikzcd}$$

• **Fixed Loop of Probability Loops** The sum of geometric distributions is a negative binomial distribution [71, p139-142]. This rule approximates such a distribution with a geometric distribution of the same mean.

$$
\begin{aligned}
&\circlearrowleft^m_n(\circlearrowleft^\rho_p(x))\colon w\\
\Rightarrow_{fps}\ &\circlearrowleft^{\rho(m-1)}_p(x)\colon w\\
&\text{where } m > 1
\end{aligned}
\tag{FP.5}
$$

Sequences of arbitrary length have a non-zero probability under both the geometric and the negative binomial distributions. Accordingly,

$$\forall u, m, \rho \bullet \mathbf{tg}(\circlearrowleft^m_n(\circlearrowleft^\rho_p(u))) = \mathbf{tg}(\circlearrowleft^\rho_p(u))$$

Fitness and precision are hence preserved. Changing probability distribution results, by definition, in a loss of stochastic information.

The $m = 1$ case is handled by Fixed loop identity CO.3.

An example use:

$$\begin{tikzcd}
\circlearrowleft^4_n\colon 5 \arrow[d] & \Rightarrow_{fps} & \circlearrowleft^6_p\colon 5 \arrow[d] \\
\circlearrowleft^2_p\colon 5 \arrow[d] & & a\colon 5 \\
a\colon 5 & &
\end{tikzcd}$$

• **Probabilistic Loop Nesting** This rule replaces a probabilistic loop of probabilistic loops as a single loop.

$$\begin{aligned} & \circlearrowleft_p^{\rho_1} (\circlearrowleft_p^{\rho_2} (x))\colon w \\ \Rightarrow_{fps} & \circlearrowleft_p^{\rho_1 \rho_2} (x)\colon w \end{aligned} \tag{FP.6}$$

Lemma 4 (Loop replacement preserves fitness)**.** $\forall u, \rho \bullet \mathbf{tg}(u) \subseteq \mathbf{tg}(\circlearrowleft_p^{\rho} (u))$.

Proof. As $\mathbf{tg}(\circlearrowleft_p^{\rho} (u))$ consists of all sequences of u and the empty trace,
$\forall u, \rho \bullet \mathbf{tg}(u) \subseteq \mathbf{tg}(\circlearrowleft_p^{\rho} (u))$ □

Extending the observation in the proof of Lemma 4, as all sequences of u and the empty trace are already represented by one probabilistic loop, $\forall u, \rho_1, \rho_2 \bullet \mathbf{tg}(\circlearrowleft_p^{\rho_1} (u)) = \mathbf{tg}(\circlearrowleft_p^{\rho_2} (\circlearrowleft_p^{\rho_1} (u))$.

Accordingly, fitness and precision are preserved.

Mixtures of geometric distributions cannot be assumed to be geometric distributions. So stochastic information is lost. The new distribution has the same mean as the original, by the product of expectations [57, p222].

An example use:
$$\begin{tikzcd} \circlearrowleft_p^{3}\colon 5 \arrow[d] & \Rightarrow_{fps} & \circlearrowleft_p^{30}\colon 5 \arrow[d] \\ \circlearrowleft_p^{10}\colon 5 \arrow[d] & & a\colon 5 \\ a\colon 5 & & \end{tikzcd}$$

• **Concurrent subsumption** Sequences already recognized as concurrent are pulled under that pattern.

$$\begin{aligned} & \times(\rightarrow(x_{1\_1}, x_{1\_2}, ...)\colon v_1, \wedge(x_{2\_1}\colon w_1, x_{2\_2}\colon w_2, ...)\colon v_2)\colon v_1 + v_2 \\ \Rightarrow_{fps} & \wedge(\Psi(x_{1\_1}\colon v_1, x_{2\_1}\colon w_1), \Gamma(\Psi(x_{1\_2}\colon v_1, x_{2\_2}\colon w_2), \frac{w_2}{v_1 + w_2}), ...)\colon v_1 + v_2 \\ & \text{where } \forall i \bullet x_{1\_i} \cong_c x_{2\_i} \end{aligned} \tag{FP.7}$$

This rule is α-stable.

By examining trace languages, we can see that sequences can be substituted into concurrency operators with similar children, with stochastic loss.

First note that concurrency across child trees always implies support for a sequence of those trees.

Lemma 5 (Sequence trace language subsets concurrency)**.**

$$\forall u_1, u_2 \in \mathcal{PPT} \bullet \mathbf{tg}(\rightarrow(u_1, u_2)\colon w_1) \subseteq \mathbf{tg}(\wedge(u_1, u_2)\colon w_2)$$

Proof. Definition 25 for sequence $\rightarrow$ and concurrency $\wedge$ shows that $\wedge(u_1, u_2)\colon w$ is a race $E_1 \,||\, E_2$ with prefix and suffix silent transitions. This race always supports, as one allowable path, one operand automaton executing completely, then the other. □

Now examining the rule, by Lemma 5, the trace language is unaffected by removing the sequence, as the traces are supported by the $\wedge$ construct. However, PPT weight consistency rules must be maintained. As the sequence is evidence the x_{1_1} subtree is more likely, we give it weight proportional to the original sequence.

$$\begin{aligned}
&\text{Substitute } u_{m1} = \Psi(x_{1\_1}\colon v_1, x_{2\_1}\colon w_1)\\
&\qquad\qquad u_{m2} = \Gamma(\Psi(x_{1\_2}\colon v_1, x_{2\_2}\colon w_2), \frac{w_2}{v_1 + w_2})\\
&\times(\rightarrow(x_{1\_1}, x_{1\_2}, ...)\colon v_1, \wedge(x_{2\_1}\colon w_1, x_{2\_2}\colon w_2, ...)\colon v_2)\colon v_1 + v_2\\
&\qquad \Rightarrow_{fps} \times(\wedge(u_{m1}, u_{m2})\colon v_1 + v_2)\colon v_1 + v_2\\
&\qquad \text{Apply single node collapse CO.1}\\
&\qquad\quad \Rrightarrow_{c} \wedge(u_{m1}, u_{m2})\colon v_1 + v_2
\end{aligned}$$

The stochastic language is not preserved, but it is approximated.

An example use:

$$\begin{tikzcd}
& & & \times:4 \arrow[dll] \arrow[drr] & & & & \Rightarrow_{fps} & & \wedge:4 \arrow[dl] \arrow[dr] & \\
& \rightarrow:1 \arrow[dl] \arrow[dr] & & & & \wedge:3 \arrow[dl] \arrow[dr] & & & a:3 & & b:1 \\
a:1 & & b:1 & & a:2 & & b:1 & & & &
\end{tikzcd}$$

- **Concurrency single node collapse** Concurrency nodes with a single child node can be collapsed with stochastic loss.

$$\wedge(u) : w \Rightarrow_{fps} u : w \tag{FP.8}$$

This rule is α-stable.

The WSFA for a concurrent operator defines preceding and succeeding silent activities, which when removed, lose stochastic information.

$$\begin{aligned}
\mathbf{wa}(\wedge(u)\colon w) &= \bigcirc \xrightarrow{\tau[w]} \bigcirc \overset{\mathbf{wa}(u)}{\dashrightarrow} \bigcirc \xrightarrow{\tau[w]} \bigcirc\\
&\Rightarrow_{fps} \mathbf{wa}(u)
\end{aligned}$$

An example use:

$$\begin{tikzcd}
\wedge:2 \arrow[d] & \Rightarrow_{fps} & a:2 \\
a:2 & &
\end{tikzcd}$$

- **Concurrency flattening** Nested concurrency operators can be flattened with loss of stochastic information.

$$\begin{aligned}
&\wedge(x_1\colon w_1, \wedge(x_2\colon w_2, x_3\colon w_3)\\
\Rightarrow_{fps}\ &\wedge(\Gamma(x_1\colon w_1, \frac{w_T + 1}{w_T}), \Gamma(x_2\colon w_2, \frac{w_T - w_1}{w_T}), \Gamma(x_3\colon w_3, \frac{w_T - w_1}{w_T})\\
&\text{where } w_T = \sum w_i
\end{aligned} \tag{FP.9}$$

This reallocates probability mass according to SLPN semantics at the child and grandchild level but may not preserve stochastic information at further levels, as discussed earlier in Section 4.3.3. Fitness and precision are preserved, together with the trace language, as previously shown for the control-flow process trees operator [96]. This rule is α-stable.

An example use:

$$\begin{tikzcd}
& \wedge\colon 4 \arrow[dl] \arrow[dr] & & \Rightarrow_{fps} & & \wedge\colon 4 \arrow[dl] \arrow[d] \arrow[dr] & \\
a\colon 1 & & \wedge\colon 3 \arrow[dl] \arrow[dr] & & a\colon 1 & b\colon 2 & c\colon 1 \\
& b\colon 2 & & c\colon 1 & & &
\end{tikzcd}$$

- **Silent sequence** Silent transitions are removed from sequences.

$$\begin{aligned}
&\rightarrow(u_1, ..., \tau, ..., u_m)\colon w \\
\Rightarrow_{fps} &\rightarrow(u_1, ..., u_m)\colon w
\end{aligned} \tag{FP.10}$$

The stochastic loss of information in this otherwise straightforward silence elimination rule happens in concurrent trees, and is discussed in Section 4.3.3. The corresponding reduction rule for control-flow process trees has no loss [96]. This rule is α-stable.

An example use:

$$\begin{tikzcd}
& \rightarrow\colon 1 \arrow[dl] \arrow[d] \arrow[dr] & & \Rightarrow_{fps} & \rightarrow\colon 1 \arrow[d] \arrow[dr] & \\
a\colon 1 & \tau\colon 1 & c\colon 1 & & a\colon 1 & c\colon 1
\end{tikzcd}$$

- **Silent concurrency** Weight from a removed silent child is redistributed to remaining nodes.

$$\begin{aligned}
&\wedge(u_1\colon w_1, ..., u_m\colon w_m, \tau\colon v)\colon w + v \\
\Rightarrow_{fps} &\wedge(\Gamma(u_1\colon w_1, \frac{w+v}{w}), ..., \Gamma(u_m\colon w_m, \frac{w+v}{w}))\colon w + v
\end{aligned} \tag{FP.11}$$

The stochastic loss of information in this otherwise straightforward silence elimination rule happens in concurrent trees, and is discussed in Section 4.3.3. The corresponding reduction rule for control-flow process trees has no loss [96]. This rule is α-stable.

As it is recognised stochastic information will be lost, it is sufficient to note that by definition, a silent activity does not change the trace language of the model, that is,

$$\mathbf{tg}(\wedge(u_1\colon w_1, ..., u_m\colon w_m, \tau\colon v)\colon w + v) = \mathbf{tg}(\wedge(\Gamma(u_1\colon w_1, \frac{w+v}{w}), ..., \Gamma(u_m\colon w_m, \frac{w+v}{w}))\colon w + v)$$

An example use:

$$\begin{tikzcd}
& \wedge\colon 6 \arrow[dl] \arrow[d] \arrow[dr] & & & \wedge\colon 6 \arrow[d] \arrow[dr] & \\
b\colon 2 & c\colon 2 & \tau\colon 2 & \Rightarrow_{fps} & b\colon 3 & c\colon 3
\end{tikzcd}$$

4.5.4 Fitness-Preserving Lossy Reductions

These rules preserve trace fitness of the input model with respect to a given log, but may reduce precision and stochastic information. They are denoted with $\Rightarrow_{fs}$.

• **Concurrent reduction from choice sequence prefixes** Concurrency is inferred when permutations of a given two-step sequence are seen at the head of a sequence. Generalizing concurrency involves re-scaling the weights of the merged sub-trees.

$$\begin{aligned}
&\times(\rightarrow(u_{x1}, u_{y1}, ...tl_1...)\colon w, \rightarrow(u_{y2}, u_{x2}, ...tl_2...)\colon v)\colon w+v\\
\Rightarrow_{fs} &\rightarrow(\wedge(\Gamma(\Psi(u_{x1}\colon w, u_{x2}\colon v), \frac{w}{w+v}), \Gamma(\Psi(u_{y1}\colon w, u_{y2}\colon v), \frac{v}{w+v}))\colon w+v,\\
&ntl)\colon w+v\\
&\text{where } u_{x1} \cong_c u_{x2} \wedge u_{y1} \cong_c u_{y2}\\
\text{and } ntl = &\begin{cases} \langle\rangle \text{ , if } tl_1 = tl_2 = \langle\rangle \\ \times(\rightarrow(...tl_1...)\colon w, \tau\colon v)\colon w+v \text{ , if } tl_1 \neq \langle\rangle \wedge tl_2 = \langle\rangle \\ \times(\tau\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v \text{ , if } tl_1 = \langle\rangle \wedge tl_2 \neq \langle\rangle \\ \times(\rightarrow(...tl_1...)\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v \text{ , otherwise} \end{cases}
\end{aligned} \tag{FPL.1}$$

As concurrency $\wedge$ introduces silent transitions, we know from Section 4.3.3 that stochastic fidelity will be lost. The weights are scaled proportionally according to which subtree is first, as evidence that those activities are more likely to win the race.

Next consider fitness and the trace languages. For the head sequences $\rightarrow(u_{x1}, u_{y1})\colon w$ and $\rightarrow(u_{y2}, u_{x2})\colon v)\colon w+v$, we can see that their trace languages are subsets of the head concurrency $\wedge$ operator in the result.

$$\begin{aligned}
u_x &= \Psi(u_{x1}\colon w, u_{x2}\colon v)\\
u_y &= \Psi(u_{y1}\colon w, u_{y2}\colon v)\\
\mathbf{tg}(\rightarrow(u_{x1}, u_{y1})\colon w) &\subseteq \wedge(u_x, u_y)\colon w \text{ by Lemma 5}\\
\mathbf{tg}(\rightarrow(u_{y2}, u_{x2})\colon w) &\subseteq \wedge(u_x, u_y)\colon w \text{ by Lemma 5}
\end{aligned}$$

Looking at the construction of the new tail ntl from the original tails tl_1 and tl_2 by cases, the first case is trivial.

$$\langle\rangle \text{ , if } tl_1 = tl_2 = \langle\rangle \text{ Trivially equal trace languages (case 1)}$$

The $\times$ operator preserves operand trace languages by definition: $\mathbf{tg}(u_1) \subseteq \mathbf{tg}(\times(u_1, u_2))$. Also by definition, silence τ preserves trace languages. This covers the last three cases.

$$\begin{aligned}
\times(\rightarrow(...tl_1...)\colon w, \tau\colon v)\colon w+v &\text{ , if } tl_1 \neq \langle\rangle \wedge tl_2 = \langle\rangle \text{ (case 2)}\\
\mathbf{tg}(\rightarrow(...tl_1...)\colon w) \subseteq \mathbf{tg}(\times(\rightarrow(...tl_1...)\colon w, \tau\colon v)\colon w+v) &\text{ Definition of choice}\\
\times(\tau\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v &\text{ , if } tl_1 = \langle\rangle \wedge tl_2 \neq \langle\rangle \text{ (case 3)}\\
\mathbf{tg}(\rightarrow(...tl_2...)\colon v) \subseteq \mathbf{tg}(\times(\tau\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v) &\text{ Definition of choice}\\
\times(\rightarrow(...tl_1...)\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v &\text{ , otherwise (case 4)}\\
\mathbf{tg}(\rightarrow(...tl_1...)\colon w) \subseteq \mathbf{tg}(\times(\rightarrow(...tl_1...)\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v) &\text{ Definition of choice}\\
\mathbf{tg}(\rightarrow(...tl_2...)\colon v) \subseteq \mathbf{tg}(\times(\rightarrow(...tl_1...)\colon w, \rightarrow(...tl_2...)\colon v)\colon w+v) &\text{ Definition of choice}
\end{aligned}$$

This shows fitness is preserved.

In the special case of sequences of two leaf nodes, fitness and precision are preserved, as:

$$\begin{aligned}\mathbf{tg}(\times(\rightarrow(a,b)\colon w_1, \rightarrow(b,a)\colon w_2)\colon w_1 + w_2) &= \{\langle a,b\rangle, \langle b,a\rangle\}\\ \mathbf{tg}(\wedge(a\colon w_1, b\colon w_2)\colon w_1 + w_2) &= \{\langle a,b\rangle, \langle b,a\rangle\}\end{aligned}$$

However for many compound subtrees, precision is not preserved. This is true of choice reduction to concurrency in general.

Lemma 6 (Concurrent reduction from choice does not maintain precision)**.** *This applies to rules of the following form:*

$$\begin{aligned}&\times(\rightarrow(u_{x1}, u_{y1})\colon w_1, \rightarrow(u_{y2}, u_{x2}\colon w_2)\colon w_1 + w_2\\ \Rightarrow_{fs}\ &\wedge(\Psi(u_{x1}, u_{x2})\colon w_3, \Psi(u_{y1}, u_{y2})\colon w_4)\colon w_3 + w_4\end{aligned}$$

Proof. Counter-example:

$$\begin{aligned}\mathbf{tg}(\times(\rightarrow(\rightarrow(a,b)\colon 1, c)\colon 1, \rightarrow(c, \rightarrow(a,b)\colon 2)\colon 2)\colon 3) &= \{\langle a,b,c\rangle, \langle c,a,b\rangle\}\\ \mathbf{tg}(\wedge(\rightarrow(a,b)\colon 1, c\colon 2)\colon 3) &= \{\langle a,b,c\rangle, \langle a,c,b\rangle, \langle c,a,b\rangle\}\end{aligned}$$

Here single node sequences have been collapsed, per rule CO.1, for clarity. An additional trace in the latter trace language results in loss of precision. □

Together with other concurrent rules such as FP.7 and FP.9, concurrency across sequences longer than two entries can be recognised, so long as all permutations are represented. As all activities are already present in both children before the application of the rule, concurrent reduction is α-stable.

An example with two leaf nodes is below. The empty tails of the sequences are omitted, and the resulting single child choice operator is again collapsed per CO.1.

An example use:

$$\begin{tikzcd}
 & & & \times\colon 3 \arrow[dll] \arrow[drr] & & & & \Rightarrow_{fs} & & \wedge\colon 3 \arrow[dl] \arrow[dr] & \\
 & \rightarrow\colon 2 \arrow[dl] \arrow[dr] & & & & \rightarrow\colon 1 \arrow[dl] \arrow[dr] & & & a\colon 2 & & b\colon 1 \\
a\colon 2 & & b\colon 2 & & b\colon 1 & & a\colon 1 & & & &
\end{tikzcd}$$

• **Concurrent reduction from choice sequence suffixes** Concurrency is inferred when permutations of a given two-step sequence are seen at the end of a sequence. This is similar to FPL.1

above, but for sequence suffixes.

$$\begin{aligned}
&\times(\rightarrow(...hd_1..., u_{x1}, u_{y1})\colon w, \rightarrow(...hd_2..., u_{y2}, u_{x2})\colon v)\colon w+v \\
\Rightarrow_{fs} &\rightarrow(nhd, \\
&\wedge(\Gamma(\Psi(u_{x1}\colon w, u_{x2}\colon v), \frac{w}{w+v}), \Gamma(\Psi(u_{y1}\colon w, u_{y2}\colon v), \frac{v}{w+v}))\colon w+v, \\
&\text{where } u_{x1} \cong_c u_{x2} \wedge u_{y1} \cong_c u_{y2} \text{ and} \\
&nhd = \begin{cases} \langle\rangle & \text{if } hd_1 = hd_2 = \langle\rangle \\ \times(\rightarrow(...hd_1...)\colon w, & \rightarrow(...hd_2...)\colon v)\colon w+v)\colon w+v \\ & \text{if } |hd_1| \geqslant 1 \wedge |hd_2| \geqslant 1 \end{cases}
\end{aligned} \tag{FPL.2}$$

The rule does not apply when only a single sequence prefix is reduced to the empty sequence. A rule along these lines does exist, but was not included as it is not a determinism-trap.

As concurrency $\wedge$ introduces silent transitions, we know from Section 4.3.3 that stochastic fidelity will be lost. The weights are scaled proportionally according to which subtree is first by sequence, as evidence that those activities are more likely to win the race.

Next consider fitness and the trace languages. Inspecting the tail sequences $\rightarrow(u_{x1}, u_{y1})\colon w$ and $\rightarrow(u_{y2}, u_{x2})\colon v)\colon w+v$, we can see that their trace languages are subsets of the tail concurrency $\wedge$ operator in the result.

$$\begin{aligned}
u_x &= \Psi(u_{x1}\colon w, u_{x2}\colon v) \\
u_y &= \Psi(u_{y1}\colon w, u_{y2}\colon v) \\
\mathbf{tg}(\rightarrow(u_{x1}, u_{y1})\colon w) &\subseteq \wedge(u_x, u_y)\colon w \text{ by Lemma 5} \\
\mathbf{tg}(\rightarrow(u_{y2}, u_{x2})\colon w) &\subseteq \wedge(u_x, u_y)\colon w \text{ by Lemma 5}
\end{aligned}$$

Looking at the construction of the new head *nhd* from the original heads hd_1 and hd_2 by cases, the first case is trivial.

$$\langle\rangle \text{ , if } hd_1 = hd_2 = \langle\rangle \text{ Trivially equal trace languages}$$

The $\times$ operator preserves operand trace languages by definition: $\mathbf{tg}(u_1) \subseteq \mathbf{tg}(\times(u_1, u_2))$. Also by definition, silence τ preserves trace languages. This covers the remaining case.

$$\begin{aligned}
&\times(\rightarrow(...hd_1...)\colon w, \rightarrow(...hd_2...)\colon v)\colon w+v)\colon w+v \\
\mathbf{tg}(\rightarrow(...hd_1...)\colon w) &\subseteq \mathbf{tg}(\times(\rightarrow(...hd_1...)\colon w, \rightarrow(...hd_2...)\colon v)\colon w+v)\colon w+v) \text{ Definition of choice} \\
\mathbf{tg}(\rightarrow(...hd_2...)\colon w) &\subseteq \mathbf{tg}(\times(\rightarrow(...hd_1...)\colon w, \rightarrow(...hd_2...)\colon v)\colon w+v)\colon w+v) \text{ Definition of choice}
\end{aligned}$$

So fitness is preserved. As this rule is a form of choice reduction to concurrency, by Lemma 6, precision is not preserved. The initial symbols of the PPT are unchanged, so it is straightforward to see the rule is α-stable. Then for the case where $hd_1 = hd_2 = \langle\rangle$ the reasoning is the same, but the initial choice can be eliminated.

An example use:

$$\begin{tikzcd}
& & \times{:}\,3 \arrow[dl] \arrow[drr] & & & & \Rightarrow_{fs} & & \to{:}\,3 \arrow[dl] \arrow[dr] & & \\
& \to{:}\,2 \arrow[dl] \arrow[d] \arrow[dr] & & & \to{:}\,1 \arrow[dl] \arrow[d] \arrow[dr] & & & \times{:}\,3 \arrow[d] \arrow[dr] & & \wedge{:}\,3 \arrow[d] \arrow[dr] & \\
a{:}\,2 & b{:}\,2 & c{:}\,2 & d{:}\,1 & c{:}\,1 & b{:}\,1 & & a{:}\,2 & d{:}\,1 & b{:}\,2 & c{:}\,1
\end{tikzcd}$$

- **Geometric Abstraction** This rule combines fixed loops into a single probabilistic loop.

$$\begin{aligned}
&\times(\circlearrowright_n^{m_1}(u){:}\, w_1, ..., \circlearrowright_n^{m_\nu}(u){:}\, w_\nu){:}\, v \\
\Rightarrow_{fs}\ &\mu(\circlearrowright_n^{m_1}(u){:}\, w_1, ..., \circlearrowright_n^{m_\nu}(u){:}\, w_\nu)
\end{aligned} \tag{FPL.3}$$

Consider $\times(\circlearrowright_n^{m_1}(u_1){:}\, w_1, ..., \circlearrowright_n^{m_n}(u_\nu){:}\, w_\nu){:}\, v$ where $\forall i, j \leqslant \nu \blacksquare u_i \cong_c u_j$. By definition of $\times$, $v = \Sigma_1^\nu w_i$. These loops are used as samples in a geometric probability distribution.

$$\text{Probability of exit } P = \frac{\Sigma_1^\nu w_i}{\Sigma_1^\nu m_i w_i}$$

$$\text{Mean repetitions } \bar{\rho} = \frac{1}{P} = \frac{\Sigma_1^\nu m_i w_i}{\Sigma_1^\nu w_i}$$

Helper function μ averages ν loops using a scaled fold with Ψ

$$\begin{aligned}
\bar{u} = \mu(\circlearrowright_n^{m_1}(u_1){:}\, w_1,..., \circlearrowright_n^{m_\nu}(u_\nu){:}\, w_\nu) = \\
\circlearrowright_p^{\bar{\rho}}\,(\Gamma(\Psi(\Gamma(u_1, m_1), \Psi(..., \Gamma(u_\nu, m_\nu))), P){:}\, v
\end{aligned}$$

By definition, $\forall \circlearrowright_n^i(u) \blacksquare \mathbf{tg}(\circlearrowright_n^i(u)) \subset \mathbf{tg}(\circlearrowright_p^\rho\,(u))$, i.e., a fixed length sequence is a strict subset of the set of all sequences. So this rule preserves fitness. However precision and stochastic information are not preserved. As an example, consider the model $\times(\circlearrowright_n^2(b{:}\,1){:}\,1, \circlearrowright_n^4(b{:}\,1){:}\,1){:}\,2$. When transformed to $\circlearrowright_p^3\,(b{:}\,2){:}\,2$, precision decreases, as the language of the model increases to include traces such as $\langle b, b, b\rangle$. Stochastic information is also lost, by summarising the individual trace lengths with a single mean.

Geometric abstraction is α-stable.

An example use:

$$\begin{tikzcd}
& \times{:}\,2 \arrow[dl] \arrow[dr] & & \Rightarrow_{fs} & \circlearrowright_p^3{:}\,2 \arrow[d] \\
\circlearrowright_n^2{:}\,1 \arrow[d] & & \circlearrowright_n^4{:}\,1 \arrow[d] & & b{:}\,2 \\
b{:}\,1 & & b{:}\,1 & &
\end{tikzcd}$$

- **Probability Loop of Fixed Loops** Sequences which repeat according to a geometric distribution can be represented more simply as a geometric distribution of the repeated tree, with the same mean.

$$\begin{aligned}
&\circlearrowright_p^\rho\,(\circlearrowright_n^m(x{:}\, w)){:}\, w \\
\Rightarrow_{fs}\ &\circlearrowright_p^{m\rho}\,(x{:}\, w){:}\, w
\end{aligned} \tag{FPL.4}$$

As the fixed loop is removed, the property of only repeating in multiples of m is lost in this transformation. Sequences of arbitrary length have a non-zero probability under the geometric distribution the transformation results in. Accordingly:
$\forall u, m, \rho_1, \rho_2 \bullet \mathbf{tg}(\circlearrowleft_p^{\rho_1} (\circlearrowleft_n^m (x\colon w))\colon w) \subseteq \mathbf{tg}(\circlearrowleft_p^{\rho_2} (u))$

Fitness is preserved.

As new sequences are introduced not of the fixed loop length, precision is lost. Consider $\circlearrowleft_p^4 (\circlearrowleft_n^2(a\colon 1))\colon 5 \Rightarrow_{fps} \circlearrowleft_p^8 (a\colon 5)\colon 5$, where the original model only accepts sequences of a where the length is a multiple of two. Counter-example trace $\sigma_e = \langle a, a, a \rangle$ then shows loss of precision, since $\sigma_e \in \mathbf{tg}(\circlearrowleft_p^8 (a\colon 5)\colon 5)) \wedge \sigma_e \notin \mathbf{tg}(\circlearrowleft_p^4 (\circlearrowleft_n^2(a\colon 1))\colon 5)$.

Changing probability distribution also results, by definition, in a loss of stochastic information.

This rule is α-stable.

An example use:
$$\begin{tikzcd}
\circlearrowleft_p^4\colon 5 \arrow[d] & \Rightarrow_{fs} & \circlearrowleft_p^{40}\colon 5 \arrow[d] \\
\circlearrowleft_n^{10}\colon 5 \arrow[d] & & a\colon 5 \\
a\colon 5 & &
\end{tikzcd}$$

• **Loop Similarity Rules** Loops are not similar to their subtrees by $\cong_c$. However loops and their children can be usefully consolidated with some loss of information. Noting that $\circlearrowleft_n^1(u)\colon w \overset{\sim}{\Longleftrightarrow} u\colon w$, we define loop similarity $\cong_L \colon \mathcal{PPT} \leftrightarrow \mathcal{PPT}$:

$$u_1 \cong_L \circlearrowleft_p^\rho (u_2) \Longleftrightarrow u_1 \cong_c u_2$$

Consider a loop rule replacement function $ls\colon (\mathcal{PPT} \to \mathcal{PPT}) \to (\mathcal{PPT} \to \mathcal{PPT})$. A reduction rule tr using $\cong_c$ has a loop-similar variation $ls \circ tr$ using $\cong_L$ and replacement tree $\bar{u}$.

$$\begin{aligned} &\text{For rule parameters } x_1\colon w_1 \cong_L \circlearrowleft_p^\rho (x_2\colon w_2)\colon w_2 \\ &\circlearrowleft_p^{\rho'} (\bar{u}) = \mu(\circlearrowleft_p^1 (x_1\colon w_1), \circlearrowleft_p^\rho (x_2\colon w_2)) \end{aligned} \tag{FPL.5}$$

The consolidated tree $\bar{u}$ may replace u_1 and u_2 in a transformation rule tr where $u_1 \cong_L u_2$ and the resulting rule application still results in tree size reduction.

Lemma 7 (Loop replacement of child preserves fitness)**.** *Take PPTs $u, u_L \in \mathcal{PPT}$ where u_L is constructed by replacing one node u_i in u with $\circlearrowleft_p^\rho (u_i)$ for some ρ. Then $\forall u \in \mathcal{PPT} \bullet \mathbf{tg}(u) \subseteq \mathbf{tg}(u_L)$.*

Proof. By cases. For $u \in \mathcal{PPT}$:

If u is a leaf $x\colon w$ where $x \in A \cup \{\tau\}$, by Lemma 4, $\mathbf{tg}(x\colon w) \subseteq \mathbf{tg}(\circlearrowleft_p^\rho (x\colon w))$.

If u is a loop $u = \circlearrowleft_p^\rho (u_i)$ and child u_i is replaced, then $\mathbf{tg}(\circlearrowleft_p^\rho (u_i))$ already represents every possible repetition of u_i, so $\mathbf{tg}(\circlearrowleft_p^\rho (u_i)) = \mathbf{tg}(\circlearrowleft_p^\rho (\circlearrowleft_p^\rho (u_i)))$.

If u is a sequence, choice or concurrency node $u = \oplus(..., u_i, ...)$ and child u_i is replaced, note again by Lemma 4 that $\mathbf{tg}(u_i) \subseteq \mathbf{tg}(\circlearrowleft_p^\rho (u_i))$. We can then reason that because the traces of u_i are all represented after the transformation, $\mathbf{tg}(\oplus(..., u_i, ...)) \subseteq \mathbf{tg}(\oplus(..., \circlearrowleft_p^\rho (u_i), ...))$ where $\oplus \in \{\to, \times, \wedge\}$.

Finally $\circlearrowleft_n$ can be considered a special case of a sequence, exhausting the cases. □

If the original rule preserves fitness, the loop-similar rule also preserves fitness. Let $tr \in \mathcal{PPT} \to \mathcal{PPT}$ be such a rule and $u \in \mathcal{PPT}$. Let V be the set of subtrees in $ls \circ tr(u)$ where a node u_i was replaced with $\circlearrowright_p^\rho\ (u_i)$. By fitness preservation Lemma 4 $\mathbf{tg}(u_i) \subseteq \mathbf{tg}(\circlearrowright_p^\rho\ (u_i))$. By Lemma 7 fitness is preserved when a child node is replaced with a loop equivalent. Accordingly, $\mathbf{tg}(u) \subseteq \mathbf{tg}(tr(u)) \subseteq \mathbf{tg}(\circlearrowright_p^\rho\ (ls \circ tr(u)))$, and fitness is preserved.

Precision is lost. Consider the application of choice folding FP.2 combined with loop similarity on the model $\times(\to(\circlearrowright_p^3\ (a\colon 1), b)\colon 1, \to(a, c)\colon 1)\colon 2$. When transformed to $\to(\circlearrowright_p^2\ (a\colon 2)\colon 2, \times(b\colon 1, c\colon 1))$, precision is lost, as traces such as $\langle a, a, a, c\rangle$, which were previously invalid, are now permitted by the model. Stochastic information is also lost.

Accordingly, if the original rule is of at least Fitness-Preserving Lossy ($\Rightarrow_{fs}$) preservation level, the final rule will be Fitness-Preserving Lossy. If the original rule is Simplifying Lossy ($\Rightarrow_s$), so is the final rule.

Loop similarity is α-stable, but note the rule it is applied to may have a weaker determinism category, which will become the category of the final rule.

4.5.5 Simplifying Lossy Reductions

The last rule category abstracts or summarises a PPT, from both a control flow and stochastic perspective, at the expense of control-flow fitness and precision. Such rules are useful for the management of noise and for allowing a user to tune the detail of the model for their specific use case. They are denoted with $\Rightarrow_s$.

- **Concurrent similarity reduction** Pairs of similar concurrent subtrees reduce to repetition.

$$\begin{aligned} &\wedge(x_1\colon w, x_2\colon v)\colon w+v \\ \Rightarrow_s\ &\circlearrowright_n^2(\Psi(x_1\colon w, x_2\colon v))\colon w+v) \text{ where } x_1 \cong_c x_2 \end{aligned} \tag{SL.1}$$

This rule is α-stable.

Consider the PPT:

$$\begin{tikzcd}
 & & & \wedge\colon 4 \arrow[dll] \arrow[drr] & & & \\
 & \to\colon 1 \arrow[dl] \arrow[dr] & & & & \to\colon 3 \arrow[dl] \arrow[dr] & \\
a\colon 1 & & b\colon 1 & & a\colon 3 & & b\colon 3
\end{tikzcd}$$

This allows the trace $\langle a, a, b, b\rangle$, therefore fitness and precision are not, in general, preserved.

An example use:

$$\begin{tikzcd}
 & \wedge\colon 4 \arrow[dl] \arrow[dr] & & \Rightarrow_s & \circlearrowright_n^2\colon 4 \arrow[d] \\
a\colon 1 & & a\colon 3 & & a\colon 4
\end{tikzcd}$$

- **Choice Pruning** Low probability paths may be pruned.

$$\begin{aligned} &\times(x\colon w_1, y\colon w_2)\colon v \Rightarrow_s x\colon v \\ &\quad \text{where } \frac{w_2}{v} < \epsilon, \text{ a supplied probability threshold} \end{aligned} \tag{SL.2}$$

Neither fitness nor precision are preserved. Consider the example:

$$\times(a\colon 99, b\colon 1)\colon 100 \Rightarrow_s a\colon 100$$

The final model loses the trace $\langle b \rangle$, so fitness is lost for logs such as $L_c = [\langle a \rangle^1]$. If the comparison log $L_{cf} = [\langle a \rangle^{10}]$, precision is increased; if it is $L_{cp} = [\langle b \rangle^{10}]$, precision is decreased, so precision cannot be preserved in general.

This rule is α-reducing, as the set of start symbols shrinks.

An example use with $\epsilon = 0.02$:

$$\begin{tikzcd}
 & & \times\colon 100 \arrow[dl] \arrow[d] \arrow[dr] & \\
 & \to\colon 95 \arrow[dl] \arrow[d] \arrow[dr] & f\colon 4 & g\colon 1 \\
a\colon 95 & b\colon 95 & c\colon 95 &
\end{tikzcd}
\Rightarrow_s
\begin{tikzcd}
 & & \times\colon 100 \arrow[dl] \arrow[dr] & \\
 & \to\colon 95.957 \arrow[dl] \arrow[d] \arrow[dr] & & f\colon 4.042 \\
a\colon 95.957 & b\colon 95.957 & c\colon 95.957 &
\end{tikzcd}$$

4.5.6 Normal Form

A normal form for a PPT is found after completely applying preserving compression rules, and using a syntactic ordering convention.

Definition 33 (Normal Form for PPTs)**.** *The normal form comprises:*

1. *No single children for non-loops (CO.1).*
2. *Flatness (CO.2).*
3. *Fixed loops are rolled (CO.4), de-nested (CO.5) and single iteration loops are removed (CO.3).*
4. *Lexical ordering for $\times$ and $\wedge$ nodes, which are commutative.*

$$\begin{tikzcd}
 & & \wedge\colon 6 \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & & \\
b\colon 1 & \tau\colon 3 & a\colon 1 & \to\colon 1 \arrow[dl] \arrow[d] \arrow[dr] & \\
 & & \to\colon 1 \arrow[dl] \arrow[d] & b\colon 1 & \tau\colon 1 \\
 & c\colon 1 & d\colon 1 & &
\end{tikzcd}
\Rrightarrow_c
\begin{tikzcd}
 & & \wedge\colon 6 \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & & & \\
a\colon 1 & b\colon 1 & \tau\colon 3 & \to\colon 1 \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
 & & c\colon 1 & d\colon 1 & b\colon 1 & \tau\colon 1
\end{tikzcd}$$

Figure 4.13: Example transformation to PPT normal form.

An example conversion to normal form is shown in Figure 4.13.

```
loopGeo :: (Eq a, Ord a) => PRule a
loopGeo = choiceChildMR loopGeoList

loopGeoList :: (Eq a) => LRule a
loopGeoList ((Node1 op1 u1 r1 w1):(Node1 op2 u2 r2 w2):ptl)
    | u1 =~= u2 = loopGeoList  (
                    Node1 PLoop (merge u1 u2)
                                (((r1*w1)+(r2*w2))/(w1+w2))
                                (w1+w2)
                    :ptl)
    | otherwise = Node1 op1 u1 r1 w1 :
                    loopGeoList ( Node1 op2 u2 r2 w2 :ptl)
...
loopGeoList pt = pt
```

Figure 4.14: Haskell code for Geometric abstraction FPL.3

4.6 Evaluation

An implementation of *Toothpaste Miner* was evaluated against existing stochastic process discovery techniques using real-world logs[2]. Quality was evaluated using the Earth Movers' Distance [95] and Alpha-precision [59] measures, simplicity and execution time.

4.6.1 Implementation

Toothpaste Miner was implemented in Haskell, exploiting the pattern-matching capabilities of that language. Haskell allows for concise expression of transformation rules, as in the code listing in Figure 4.14. PPT similarity $\cong_c$ is implemented as the `=~=` operator.

This implementation includes discovery, and SLPN conversion to PNML. It maintains $\wedge$ and $\times$ nodes in lexical order for cheaper comparisons, and to limit traversal distance for similarity rules such as *Choice similarity* FP.1. Concurrency identification is limited to sequences of length two. The implementation outputs PPT models in the normal form described in Section 4.5.6. Noise reduction is implemented by adding the Choice Pruning Rule SL.2 to the ruleset after a full discovery run without that rule. XES log parsing was performed by a Python script.

Two cycles of implementation and evaluation were conducted. The first used a binary tree implementation of PPTs. This chapter reports on the second cycle of evaluation, with a multi-node implementation, as at version v0.9.3.0.

[2]Source code for the implementation, experiment setup and result files are all available at `https://github.com/adamburkegh/toothpaste`. See Appendix A

4.6.2 Evaluation Design

In order to evaluate the practical use of our discovery technique, it was compared to established stochastic discovery techniques in the literature with public implementations. K-fold cross validation ($k = 5$) was used on six logs, and results compared using stochastic quality criteria, simplicity and computation time, summarized in Table 4.3.

Table 4.3: Discovery Evaluation Design

Logs	Techniques	Measures
BPIC 2013 closed [141]	Toothpaste *dtm* (Definition 28)	XPU [59] (and Chapter 6)
BPIC 2018 control [61]	GDT_SPN Discovery [130]	Entity count
BPIC 2018 reference [61]	walign-inductive (Chapter 3)	Execution time
Road Traffic Fines [105]	wfreq-split (Chapter 3)	EM [95]
Sepsis [111]	wpairscale-split (Chapter 3)	
Teleclaims [7]	Trace model	

The Trace Model benchmark is constructed with unity weights. The estimators *walign-inductive*, *wfreq-split* and *wpairscale-split* combine a control-flow discovery step, in this case the Split or Inductive miners, with a weight estimation step (see Chapter 3). These combinations were chosen based on performance in previous experiments in the literature, described in Section 4.1.

The Existential precision measure, XPU, is an adaptation of the Alpha-precision measure [59]. The Alpha Precision measure uses the stochastic language of the model and the event log to make inferences about the underlying system that generated the log. This depends on a parameter called alpha significance which varies across domains, making the comparison of models across logs difficult. We use the Existential Precision metric (XPU) as an alternative to allow such comparisons. We introduce XPU in Section 6.3.5 in Chapter 6, with other conformance metrics.

As the models output by Toothpaste discovery can be non-deterministic, that is at certain stages multiple externally indistinguishable steps can be taken, measures based on entropy over SDFAs [100, 126] were not used in this evaluation.

4.7 Results and Discussion

We review the experimental results of our evaluation in Section 4.7.1. Section 4.7.2 discusses alternative designs for the PPT formalism.

4.7.1 Experimental Results

Experiments were run on a Windows 10 machine with a 2.3GHz CPU, 10 Gb of allocated memory and Haskell Stack 2.9.1 with GHC 8.10.7. Figure 4.15 shows the performance of the miners plotting Earth movers' distance against Alpha-precision. Where results for a technique do not

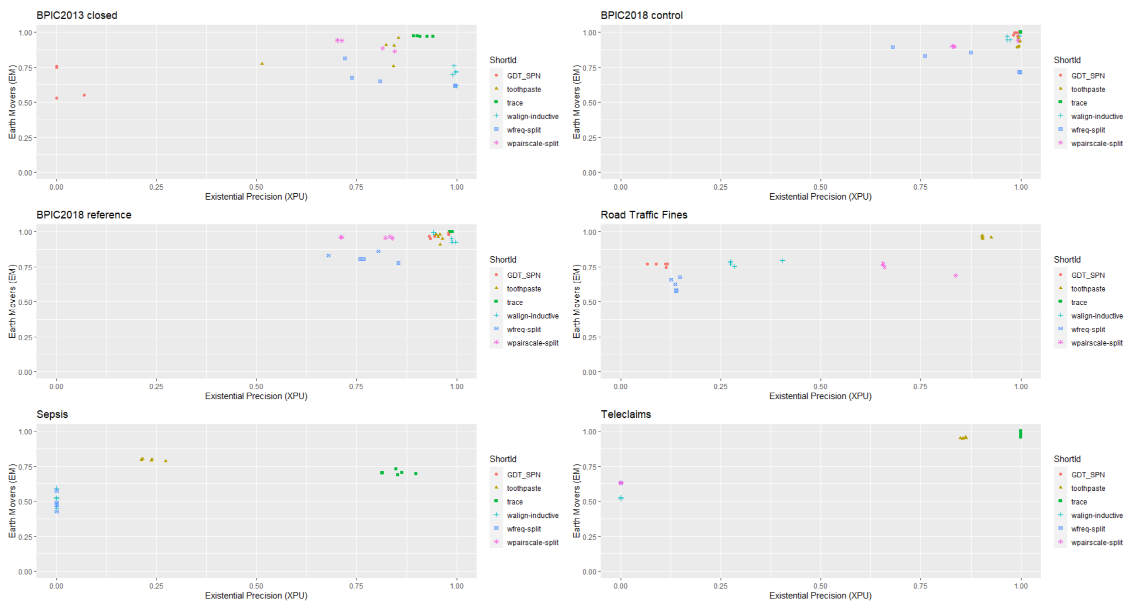


Figure 4.15: Earth Movers' Distance (EM) and Existential Precision (XPU) for stochastic process miners across six real-life or realistic logs.

appear, either no model was produced, or measures did not return a result. These were caused by either not returning within a timeout of eight hours, or exceeding available memory.

The Toothpaste Miner models are able to consistently adhere to the process described by the logs across a number of domains, at an Earth-Movers' Distance (EM) measure of 0.75 or better. Performance on the Existential precision measure (XPU) is also typically above 0.75. On EM and XPU, the Toothpaste Miner showed the best performance on the Road Traffic Fines, Sepsis and Teleclaims logs, was close to the best technique for the remainder, and able to produce models for all experimental logs used. On the challenging Sepsis log, which is the record of a hospital treatment process, precision is low at 0.25, but also the highest of any of the discovery techniques. To analyse the interaction of EM, Alpha-precision and model entity count, Pareto frontiers of mean measures by technique were constructed for each log. Toothpaste is on the frontier for the BPIC 2013 closed, Road Traffic Fines, Sepsis, and Teleclaims logs.

When unable to apply reduction rules, Toothpaste Miner will return a complicated model rather than no model. This is in keeping with a general trade-off of consistency and quality against simplicity, as seen in Table 4.4. PPTs allow a more compact representation than Petri nets. For example a sequence will have nearly twice the places and transitions of its PPT equivalent, and a loop even more. However, for a fair comparison, we use SLPNs as the standardised representation, and the high number of entities for some models is in line with the more complicated models produced on some logs. In these cases the Toothpaste models can act as a quality-preserving form of log compression, as determined by the rules used during discovery. They can also be used as part of an analysis pipeline, rather than as standalone human-readable diagrams.

Toothpaste executed in 2-11 seconds across the range of logs, including the execution of the Python log conversion script, showing the practical feasibility of the technique.

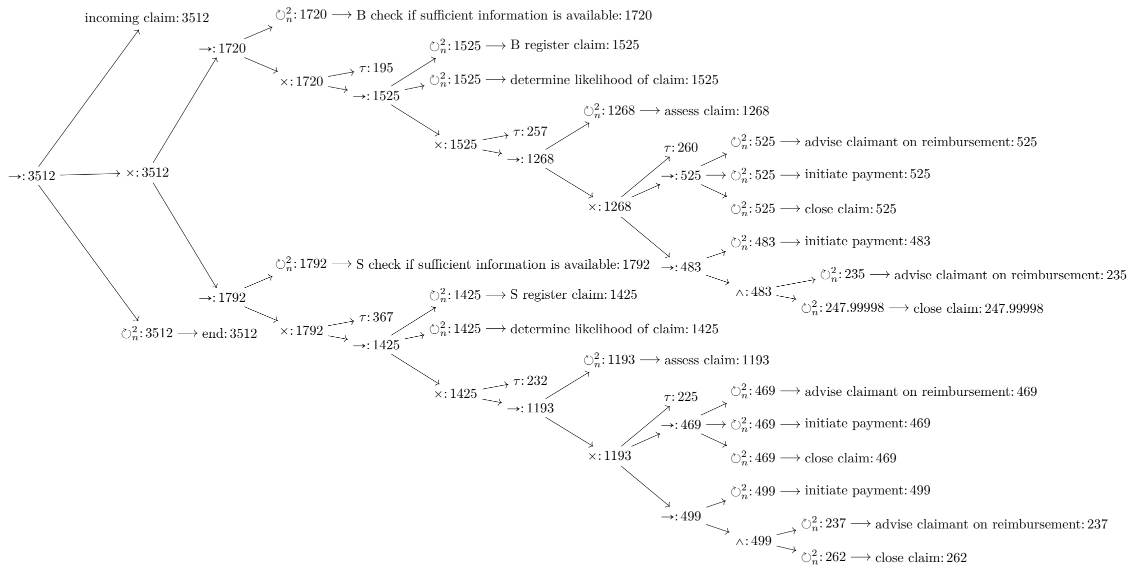

incoming claim: 3512
→: 3512
×: 3512
⟳: 3512 ⟶ end: 3512
→: 1720
⟳: 1720 ⟶ B check if sufficient information is available: 1720
×: 1720
τ: 195
→: 1525
⟳: 1525 ⟶ B register claim: 1525
⟳: 1525 ⟶ determine likelihood of claim: 1525
×: 1525
τ: 257
→: 1268
⟳: 1268 ⟶ assess claim: 1268
×: 1268
τ: 260
→: 525
⟳: 525 ⟶ advise claimant on reimbursement: 525
⟳: 525 ⟶ initiate payment: 525
⟳: 525 ⟶ close claim: 525
→: 483
⟳: 483 ⟶ initiate payment: 483
∧: 483
⟳: 235 ⟶ advise claimant on reimbursement: 235
⟳: 247.99998 ⟶ close claim: 247.99998
→: 1792
⟳: 1792 ⟶ S check if sufficient information is available: 1792
×: 1792
τ: 367
→: 1425
⟳: 1425 ⟶ S register claim: 1425
⟳: 1425 ⟶ determine likelihood of claim: 1425
×: 1425
τ: 232
→: 1193
⟳: 1193 ⟶ assess claim: 1193
×: 1193
τ: 225
→: 469
⟳: 469 ⟶ advise claimant on reimbursement: 469
⟳: 469 ⟶ initiate payment: 469
⟳: 469 ⟶ close claim: 469
→: 499
⟳: 499 ⟶ initiate payment: 499
∧: 499
⟳: 237 ⟶ advise claimant on reimbursement: 237
⟳: 262 ⟶ close claim: 262

Figure 4.16: Model mined by Toothpaste *dtm* from the teleclaims log.

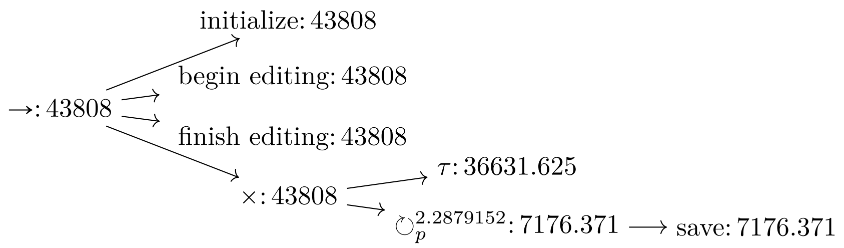


Figure 4.17: Model mined by Toothpaste *dtm* from the BPIC2018 control log with 0.1 noise reduction.

Log	Miner	Execution time (ms)		Entity Count		EM	XPU
		Mean	Std dev	Mean	Std dev	Mean	Mean
BPIC2013 closed	GDT_SPN	181.00	251.57	25.20	3.90	0.67	0.01
BPIC2013 closed	toothpaste	2863.00	1899.89	435.00	108.20	0.86	0.78
BPIC2013 closed	trace	12.00	12.88	2368.60	105.71	0.97	0.91
BPIC2013 closed	walign-inductive	110.20	150.37	7.20	1.64	0.70	1.00
BPIC2013 closed	wfreq-split	767.00	58.37	16.00	1.41	0.67	0.85
BPIC2013 closed	wpairscale-split	770.20	64.84	16.00	1.41	0.91	0.76
BPIC2018 control	GDT_SPN	706.20	262.90	31.60	1.34	0.99	0.99
BPIC2018 control	toothpaste	9069.40	204.30	51.20	6.83	0.92	0.99
BPIC2018 control	trace	177.80	52.08	55758.80	485.53	1.00	1.00
BPIC2018 control	walign-inductive	886.00	224.45	21.80	1.30	0.96	0.98
BPIC2018 control	wfreq-split	2205.80	63.63	20.60	2.41	0.80	0.86
BPIC2018 control	wpairscale-split	2195.00	82.37	20.60	2.41	0.92	0.90
BPIC2018 reference	GDT_SPN	1177.40	582.45	32.60	1.82	0.97	0.95
BPIC2018 reference	toothpaste	9592.00	241.33	1379.80	190.92	0.96	0.96
BPIC2018 reference	trace	137.40	41.83	42663.20	311.90	1.00	0.99
BPIC2018 reference	walign-inductive	864.00	223.00	22.00	1.22	0.96	0.97
BPIC2018 reference	wfreq-split	2086.00	121.66	23.00	2.74	0.81	0.77
BPIC2018 reference	wpairscale-split	2077.40	90.65	23.00	2.74	0.96	0.78
Road Traffic Fines	GDT_SPN	61796.60	122011.90	68.80	3.77	0.76	0.10
Road Traffic Fines	toothpaste	9387.40	151.13	355.00	30.81	0.96	0.91
Road Traffic Fines	trace	657.20	155.05	194516.00	370.02	NaN	NaN
Road Traffic Fines	walign-inductive	1597.60	343.21	32.20	3.27	0.78	0.30
Road Traffic Fines	wfreq-split	2351.20	111.64	31.80	0.45	0.62	0.14
Road Traffic Fines	wpairscale-split	2350.40	99.34	31.80	0.45	0.75	0.69
Sepsis	GDT_SPN	10147434.00	11267371.16	95.00	6.48	NaN	0.00
Sepsis	toothpaste	4711.40	3224.87	3057.00	282.94	0.79	0.23
Sepsis	trace	25.00	23.58	5877.60	335.38	0.70	0.86
Sepsis	walign-inductive	580.40	152.72	50.40	4.16	0.52	0.00
Sepsis	wfreq-split	926.80	57.44	48.40	1.67	0.49	0.00
Sepsis	wpairscale-split	969.50	57.28	48.00	1.41	NaN	NaN
Teleclaims	GDT_SPN	453.00	27.14	60.00	0.00	NaN	0.00
Teleclaims	toothpaste	2404.60	329.08	116.00	0.00	0.95	0.86
Teleclaims	trace	61.60	33.31	17754.80	241.82	0.98	1.00
Teleclaims	walign-inductive	143.80	107.50	28.60	0.55	0.52	0.00
Teleclaims	wfreq-split	1267.25	710.65	56.25	0.50	0.63	0.00
Teleclaims	wpairscale-split	1192.00	645.96	56.20	0.45	0.63	0.00

Table 4.4: Summary statistics for different miners and logs.

Figure 4.16 shows a model mined from teleclaims log with zero noise reduction. This shows the identification of control constructs and the estimation of weights in a human readable model. There is a repeated subtree in this model, starting at $\times\colon 1525$, which the current rules do not identify due to the occurrence at a "grandchild" level among differing subtrees. Figure 4.17 shows a compact model mined from the BPIC2018 control log, which is a documentation process for the EU Common Agricultural Policy. A noise reduction parameter of 0.1 was used.

4.7.2 Alternative Semantics and Connections To Process Algebras

In the definition for PPTs (Definition 25) used throughout this chapter, care has been taken to maintain compatibility with SLPNs and Petri net derived semantics. This necessarily includes uses of weighted silent transitions for some control flow constructs, particularly for concurrency and probabilistic loops. This is both a feature and limitation of Petri net representation, or in other words a form of representational bias [6, p118].

The impact of silent states on composition is discussed in Section 4.3.3 and a useful example is in Figure 4.8. It is possible to define alternative loop and concurrency semantics which do not include structural silent events in the resulting weighted paths.

Compositional Concurrency operator. The $\dot{\wedge}$ operator represents parallel execution of child trees. It provides nesting consistency where $\dot{\wedge}(u_1, \dot{\wedge}(u_2, u_3)) \overset{\sim}{\Longleftrightarrow} \dot{\wedge}(u_1, u_2, u_3)$.

$$\begin{aligned}\mathbf{wa}(\dot{\wedge}(u_1, u_2, ..., u_n) \colon w) &= \mathbf{wa}(u_1) \,||\, \mathbf{wa}(u_2) \,||\, ... \,||\, \mathbf{wa}(u_n) \\ \forall s_1, .., s_n, w_1, .., w_n \blacksquare \dot{\wedge}(s_1 \colon w_1, ..., s_n \colon w_n) \colon w &\implies w = \sum_{(s_j, w_j)} w_j\end{aligned}$$

Compositional Probabilistic Loops. The probabilistic loop operator $\odot^{\rho}_{p}(u)$ executes the child tree with the probability of exiting at each iteration of $\frac{1}{\rho}$ where $\rho \in \mathbb{R}^+ \wedge \rho \geqslant 1$, and without intermediate entry or exit states. The weight of the child node is the weight of the parent loop. The exit state of the child is made the entry state, forming a loop. It supports an infinite trace language of finite traces.

$$\begin{aligned}\mathbf{wa}(\circlearrowright^{\rho}_{p} (x_1 \colon w_1) \colon w_1) &= E_{loop} \\ \text{where } E_{loop} &= (S_L, A_L, \hookrightarrow_L, s_0, s_0) \\ E_1 &= (S_1, A_1, \hookrightarrow_1, s_{1.0}, s_{1.\omega}) = \mathbf{wa}(x_1 \colon w_1) \\ S_L &= S_1 \\ A_L &= A_1 \\ \hookrightarrow_L &= \{s_0 \overset{x[w]}{\hookrightarrow} s_{1.2} \mid s_{1.1} \overset{x[v]}{\hookrightarrow}_1 s_{1.2} \wedge s_{1.2} \neq s_{1.\omega} \wedge w = \frac{v \cdot (\rho - 1)}{\rho}\} \\ &\cup \{s_{1.1} \overset{x[w]}{\hookrightarrow} s_0 \mid s_{1.1} \overset{x[v]}{\hookrightarrow}_1 s_{1.\omega} \wedge w = \frac{v \cdot (\rho - 1)}{\rho}\} \\ &\text{and for the weights of exiting transitions } TE \frac{w}{\rho} = \sum_{v \in TE} v\end{aligned}$$

Unlike other PPT constructs, this loop version imposes conditions on the weights of the exit transitions from the component WSFA, which may cause other composition difficulties.

Such definitions suggest simpler expression of some composed concurrency and loop constructs is possible. As place-transition nets are Turing-complete [124], at least if given an unlimited budget of places and transitions similar to a Turing machine's infinite tape, it seems likely that such constructs are still translatable to some form of Petri net. It is less clear that a strictly equivalent Stochastic Petri net would be constructable under stricter complexity constraints, precisely because the concurrency mechanism requires an intermediate silent state.

These alternative semantics suggest connections to stochastic process algebras such as PEPA [84] or EMPA [26]. The connection to the powerful concurrency modelling constructs in process algebras has been noted since the beginning of process mining research [8], and more recent research uses process mining discovery algorithms to construct control-flow process algebra models [23]. Process algebras have different strengths and weaknesses to Petri nets. For example: "Stochastic process algebras lack the attractive graphical presentation of Petri nets [but have] an explicit compositional structure." [74].

In PEPA specifically, the co-operation operator represents parallel execution with possible mutual communication with timing resolved by a race. Durations in general are described by negative exponential distributions.

Though the elegant expression of concurrency without additional internal states is appealing, the current investigation is more concerned with probability than the combined problems of probability and time. There are also benefits to using notations well-understood in the process mining community. Process mining using stochastic process algebras may be a useful direction for future research.

4.8 Summary

This chapter presents one set of solutions to the problem of unsupervised learning of stochastic process models. Our solution uses a novel formalism, Probabilistic Process Trees, with well-defined semantics, and this allowed us to understand the challenges of concurrency and determinism in stochastic models. Through the use of well-defined reduction and abstraction rules, we also introduced two polynomial time discovery algorithms which calculate such stochastic models, and support noise reduction. An implementation and experimental evaluation produced models, which achieve a consistently high quality across a number of real-life logs, at feasible execution times, while trading model simplicity for quality in some cases.

In Chapter 5, we show solutions on PPTs for computing probability of a trace, with an approximation bound, when using a stochastic process model.

In Chapter 6, we introduce a genetic miner discovery algorithm which works on PPT models. We also use Toothpaste Miner to discover models included in a large data set of stochastic models. This dataset is used to quantitatively analyse common properties shared across models.

Probabilistic Process Trees are designed, as a formalism, to have commonalities with formalisms in control-flow process mining. This obviously includes process trees, but also workflow nets and single-entry single-exit regions. These structures give guarantees for a number of formal properties, such as safeness and soundness. Future work demonstrating those properties, and formally articulating the connection to the theory of regions, would enable algorithms and techniques that leverage those guarantees.

Though we show a number of properties for PPT rules, this did not include confluence [27, p10], whereby when there is a choice of applicable rules, the order in which they are applied does not change the final result. This would be a desirable property to show in future work, as it is often part of term-rewriting and subtree replacement systems. The Toothpaste Miner restriction that a rule must always reduce the number of nodes in a tree may add further complications to its proof or disproof.

Future work on discovery can include algorithms which produce simpler models, algorithms with determinism guarantees, and algorithms which integrate the quality-preservation aspects of rules with other forms of structural simplification.

Chapter 5

Trace Probability With Probabilistic Process Trees

In many instances it is exceedingly difficult to estimate beforehand the sale of an article, or the effects of a machine.

Charles Babbage
On the Economy of Machinery and Manufactures [16]

The promise of stochastic process models is that they can be used for detailed and precise descriptions of probability. A particularly useful calculation is for *trace probability*: the probability that a trace will occur under a given model (Definition 13). Until recently (indeed, within the research timeframe of this thesis), this was an unsolved problem for the sorts of stochastic process models most often used for process mining, those that support concurrency, loops, silent activities, and duplicate labels. Models with these constructs are used in process mining because they occur in data from and models for real world organisations.

This chapter presents two solutions to the trace probability problem, TRACE-PROB (Definition 13). The first is on Weighted Stochastic Finite Automata (WSFAs) (Definition 21). The second, and more novel, solution, is on Probabilistic Process Trees (PPTs) (Definition 25). PPTs were introduced in Chapter 4, where they were used as the basis of discovery algorithms. Here we show their application to a problem related to process mining conformance. The solution allows precise calculation of trace probability within a given approximation bound, including for loops, duplicates and silent transitions.

Section 5.1 reviews related work. Section 5.2 describes a solution for trace probability on WSFAs, while Section 5.3 introduces a new solution to trace probability on PPTs. Section 5.4 discusses the solution and prototype implementation. Section 5.5 summarises the chapter in the context of other work in this thesis.

5.1 Related Work

Stochastic process model conformance is a recent and active research topic. Stochastic process models make it possible to answer questions of probability, including the probability that a particular sequence of activities occurs in a process. Stochastic extensions for Petri nets include Stochastic Petri Nets (SPNs) and Generalised SPNs (Definition 9). The stochastic behaviour of such nets can be captured through a discrete-time Markov chain [20]. However traditional stochastic nets do not support transition labels or silent transitions. A common focus is solving for a steady-state, describing the probabilities and behaviour as time approaches infinity. This is done through construction of discrete-time Markov chains. Process mining, by contrast, must commonly deal with silent and duplicate labels. As the models are describing individual passes through a workflow, terminating executions are of more interest, and many process mining models, such as workflow nets [6, p65], are designed to guarantee termination with all tokens at a final place.

Determining the probability that a particular sequence of activities occurs in a stochastic process model has accordingly been formulated as the Trace-Prob problem [98]. A linear programming approach [98] provides one solution to Trace-Prob on SLPNs, also addressing traces with certain automata-based constraints. Another recent solution uses an expectation-minimisation (EM) algorithm on a novel model representation based on trees and probabilistic context free grammars [157]. The work reported in Section 5.3 expands our understanding of stochastic process models by providing a new, bounded, approximation for Trace-Prob on any PPT model.

Other contributions to stochastic conformance define quality measures for stochastic process models. The Earth-movers' distance (EMD) measure [95] represents shared probability mass between models or between logs and models, building on the well-known Levenshtein string edit distance. Measures for entropy precision and recall (fitness) [101], and entropic relevance [14] have also been defined.

Entropy is typically defined using probability. Let X be a discrete random variable on the alphabet B and distributed according to $Pr\colon B \rightarrow [0,1]$.

$$H(X) = -\sum_{x \in B} Pr(x) \log_2 Pr(x)$$

The formulae for entropy precision, recall and entropic relevance take trace probability as an input for a calculation over a stochastic language. A stochastic language is a real-valued multiset of traces where the frequencies are probability values for each trace, as in Definition 2. Alternatively and equivalently, a stochastic language can be a set of trace-probability pairs. Metrics using entropy are constrained in definition and implementation to use Stochastic Deterministic Finite Automata (SDFAs) [152], because a trace probability approximation is only defined for SDFAs. By providing a solution on non-deterministic PPTs, the current chapter allows the extension of these metrics, though this is not part of our contribution.

An Alpha-precision measure [59] interprets precision in terms of the statistical significance of different paths in comparing different models and logs. Alpha-precision depends on trace probability, termed "model probability", as an input. This may have influenced the demonstration of the measure on Direct Follow Graphs rather than another formalism. The links between simplicity and entropy have also been investigated [89], emphasizing a distinction between *behavioural* simplicity and *structural* simplicity. Behavioural simplicity attempts to describe whether the behaviour of a system can be described with little information. Structural simplicity measures a particular representation.

Beyond metrics, but still within stochastic conformance, trace probability is also an input for the calculation of probabilistic alignments [24].

5.2 Trace Probability for Weighted Stochastic Finite Automata

Weighted Stochastic Finite Automata (WSFAs) are built around probabilities for transitioning between states, so have a trace probability calculation implicit in their design. However the WSFA design, like many automata, have no special treatment for silence. τ labels are treated as just another symbol. That multiple paths can result in the same trace requires special handling in the probability calculation.

We will describe two probability functions:

Definition 34 (Trace Probability and Bound Approximation for WSFAs)**.**

Let $A \subseteq \mathcal{A}$ be the activity set for the input WSFA

$\pi_{wsfa} \colon A^* \times \mathcal{WS} \to [0,1]$ *trace probability on WSFAs*

$\pi_{w\epsilon} \colon A^* \times \mathcal{WS} \times (0,1] \to [0,1]$ *trace probability with approximation bound*

The π_{wsfa} function is simpler, but is not a computable function, as there are some automata where it is defined but will not return in finite time. The $\pi_{w\epsilon}$ is a computable function which returns the trace probability accurate to an approximation bound.

5.2.1 Automata Without Silent Loops

To detail the π_{wsfa} function, we use a recursive intermediate function π_{st} that takes a particular state as input.

$$\pi_{st} \colon A^* \times \mathcal{WS} \times S \to [0,1] \text{ where } S \text{ is the set of states for the input WSFA}$$

$$\text{Let } E = (S, A \cup \{\tau\}, \hookrightarrow, s_0, s_\omega)$$

$$\begin{aligned}
\pi_{st}(\langle a \rangle + \sigma, E, s) &= \frac{1}{w_T}\sum_{T_a} w \cdot \pi_{st}(\sigma, E, s') + \frac{1}{w_T}\sum_{T_\tau} w \cdot \pi_{st}(\langle a \rangle + \sigma, E, s') \\
\pi_{st}(\langle\rangle, E, s_\omega) &= 1 \\
\pi_{st}(\langle\rangle, E, s) &= \frac{1}{w_T}\sum_{T_\tau} w \cdot \pi_{st}(\langle\rangle, E, s') \text{ where } s \neq s_\omega
\end{aligned}$$

$$\begin{aligned}
\text{where } T_a &= \{s \overset{a[w]}{\hookrightarrow} s'\} \text{ and } a \in A \\
T_\tau &= \{s \overset{\tau[w]}{\hookrightarrow} s'\} \text{ silent arcs} \\
w_T &= \sum_{s \overset{x[w]}{\hookrightarrow} s'} w
\end{aligned}$$

For an input state, π_{st} looks at all the transitions from that state. For those that match the head of the input trace, it takes the chance of that transition multiplied by the probability of the suffix subtrace. For those that are silent, it takes the chance of those multiplied by the entire input trace. For the empty trace, if the WSFA is at the terminal state, it's a match. If it is not, only silent transitions can still result in a match, and any other symbol terminates the recursion.

The overall probability function π_{wsfa} is then:

$$\pi_{wsfa}(\sigma, (S, Act, \hookrightarrow, s_0, s_\omega)) = \pi_{st}(\sigma, (S, Act, \hookrightarrow, s_0, s_\omega), s_0)$$

That is, this function adds the probability of all the possible matching paths.

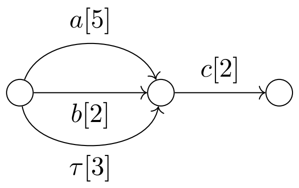


Figure 5.1: Example WSFA model, E_1.

Considering the example WSFA in Figure 5.1 against various example traces:

$$\begin{aligned}
\pi_{wsfa}(\langle a, c \rangle, E_1) &= \frac{5}{10} \cdot \frac{2}{2} + \frac{3}{10} \cdot 0 = \frac{1}{2} \\
\pi_{wsfa}(\langle c \rangle, E_1) &= \frac{3}{10} \cdot \frac{2}{2} = \frac{3}{10} \\
\pi_{wsfa}(\langle c, b \rangle, E_1) &= \frac{3}{10} \cdot \frac{2}{2} \cdot 0 = 0
\end{aligned}$$

Figure 5.2 shows why π_{wsfa} is not a computable function. The looping arc marked $\tau[1]$ will cause the π_{st} function to not terminate, as the input trace will not decrease in size on each traversal

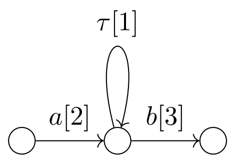


Figure 5.2: Example WSFA with silent loop, $E_{\tau loop}$.

of the loop and extra level of recursion. This is similar to the problem of establishing the stochastic language for SLPNs with silent transitions and loops [102].

5.2.2 Approximating Loops Including Silence

The problems of silence in loops can be fixed by accepting an approximation. The $\pi_{w\epsilon}$ is a total function that takes an approximation parameter. We use a new intermediate function $\pi_{st\epsilon}$ to define it. Similarly, it passes an approximation parameter and the probability of the head sequence so far inspected.

$$\pi_{st\epsilon}\colon A^{*} \times \mathcal{WS} \times S \times (0,1] \times [0,1] \times \mathbb{P}(S) \rightarrow [0,1]$$

where S is the set of states for the input WSFA

Let $E = (S, A \cup \{\tau\}, \hookrightarrow, s_0, s_\omega)$

$$\pi_{st\epsilon}(\langle a\rangle + \sigma, E, s, \epsilon, p, \mathit{seen}) = \frac{1}{w_T}\sum_{T_a} w \cdot \pi_{st\epsilon}(\sigma, E, s', \epsilon', \frac{pw}{w_T}, \mathit{seen} \cup \{s\}) + \frac{1}{w_T}\sum_{T_\tau} w \cdot \pi_{st\epsilon}(\langle a\rangle + \sigma, E, s', \epsilon', \frac{pw}{w_T}, \mathit{seen} \cup \{s\}) \text{ , if } p > \epsilon$$

$$\pi_{st\epsilon}(\langle a\rangle + \sigma, E, s, \epsilon, p, \mathit{seen}) = \frac{1}{w_T}\sum_{T_a} w \cdot \pi_{st\epsilon}(\sigma, E, s', \epsilon', \frac{pw}{w_T}, \mathit{seen} \cup \{s\}) \text{ , if } p \leqslant \epsilon$$

$$\pi_{st\epsilon}(\langle\rangle, E, s_\omega, \epsilon, p, \mathit{seen}) = 1 \text{ ; expected termination}$$

$$\pi_{st\epsilon}(\langle\rangle, E, s, \epsilon, p, \mathit{seen}) = \frac{1}{w_T}\sum_{T_\tau} w \cdot \pi_{st\epsilon}(\langle\rangle, E, s', \epsilon', \frac{pw}{w_T}, \mathit{seen} \cup \{s\}) \text{ , if } p > \epsilon \text{ and } s \neq s_\omega$$

$$\pi_{st\epsilon}(\langle\rangle, E, s, \epsilon, p, \mathit{seen}) = 0 \text{ , if } p \leqslant \epsilon \text{ and } s \neq s_\omega \text{ ; terminate low probability repeated path}$$

where $T_a = \{s \overset{a[w]}{\hookrightarrow} s'\}$, activity arcs

$T_\tau = \{s \overset{\tau[w]}{\hookrightarrow} s'\}$, silent arcs

$$w_T = \sum_{s \overset{x[w]}{\hookrightarrow} s'} w$$

$$\epsilon' = \begin{cases} \frac{w \cdot \epsilon}{w_T} & \text{where } s \in \mathit{seen} \text{ ; scale down threshold for repeated state} \\ \epsilon & \text{otherwise} \end{cases}$$

The structure of the calculation is similar to π_{st}. The first difference here is that improbable paths involving silence are treated as probability zero. To achieve this, the function keeps track of the visited states in the parameter *seen*, and the cumulative trace probability. If visiting a new state, the suppression threshold, ϵ, is scaled down, so that the overall probability error will be within the original threshold. If visiting a previously seen state, the suppression threshold is unchanged. So in a loop involving silence, the current probability will monotonically reduce, while the suppression threshold will not, eventually terminating the calculation.

The overall probability approximation function $\pi_{w\epsilon}$ is then:

$$\pi_{w\epsilon}(\sigma, (S, Act, \hookrightarrow, s_0, s_\omega), \epsilon) = \pi_{st\epsilon}(\sigma, (S, Act, \hookrightarrow, s_0, s_\omega), s_0, \epsilon, 1, \varnothing)$$

An example probability calculation uses the WSFA $E_{\tau loop}$ in Figure 5.2.

$$\begin{aligned}
\pi_{w\epsilon}(\langle b, c\rangle, \pi_{w\epsilon}, \frac{1}{10}) &= \frac{2}{2}\cdot\frac{2}{3} && \text{for path } \langle b, c\rangle \\
&+ \frac{2}{2}\cdot\frac{1}{3}\cdot\frac{2}{3} && \text{for path } \langle b, \tau, c\rangle \\
&+ \frac{2}{2}\cdot\frac{1}{3}\cdot\frac{1}{3}\cdot\frac{2}{3} && \text{for path } \langle b, \tau, \tau, c\rangle \\
&+ \frac{2}{2}\cdot\frac{1}{3}\cdot\frac{1}{3}\cdot 0\cdot\frac{2}{3} && \text{for path } \langle b, \tau, \tau, \tau, c\rangle \text{ and thereafter} \\
&= \frac{2}{3} + \frac{2}{9} + \frac{2}{27} \\
&= \frac{26}{27}
\end{aligned}$$

As $E_{\tau loop}$ supports only one valid trace, we can see that the cumulative sum should be equal to 1, but is instead approximated by $\frac{26}{27}$, with an error of $\frac{1}{27}$, within the bound of $\frac{1}{10}$.

5.3 Trace Probability for Probabilistic Process Trees

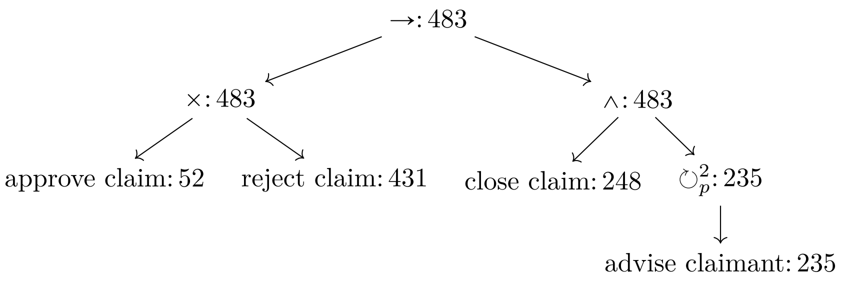


Figure 5.3: Detail of a claims process as a Probabilistic Process Tree (PPT). Repeat of Figure 4.4.

Using the example model in Figure 5.3, we might ask the probability that claimants have to be advised of a successful claim outcome exactly once. Using the techniques in this section it can be shown this is:

$$\pi_{ppt}(\langle \text{advise claimant}\rangle, \circlearrowright_p^2 (\text{advise claimant: } 235)) = \frac{1}{4}$$

We can also answer questions about longer sequences of activities and more complicated models. This section presents a bounded approximation for TRACE-PROB on PPTs, and closed form solutions for useful subsets of PPTs. These together define terminating recursive functions π_{ppt} and π_ϵ for trace probabilities on PPTs.

Definition 35 (Trace Probability and Bound Approximation for PPTs)**.** *Let $A \subseteq \mathcal{A}$ be the activities for a given PPT. Then*

$$\begin{aligned} \pi_{ppt}\colon A^* \times \mathcal{PPT} &\to [0,1] && \textit{trace probability on PPTs} \\ \pi_\epsilon\colon A^* \times \mathcal{PPT} \times (0,1] &\to [0,1] && \textit{trace probability approximation with bound} \end{aligned}$$

In the sections below, we introduce loudness and boundedness for PPTs. The π_{ppt} function is then defined by cases for leaf nodes, concurrency $\wedge$, choice $\times$, sequences $\rightarrow$, fixed loops $\circlearrowright_n$, and probabilistic loops $\circlearrowright_p$. Some PPTs with silent activities within probabilistic loops are not computable for the function π_{ppt}. In defining the solution, we take $A \subseteq \mathcal{A}$ to be the activity set for the input PPT. The π_ϵ function is a computable function, and for all $\sigma \in A^*, u \in \mathcal{PPT}, \epsilon \in [0,1] \bullet \pi_{ppt}(\sigma, u) = \pi_\epsilon(\sigma, u, \epsilon)$ so long as π_{ppt} terminates.

5.3.1 Loud Trees and τ-Boundedness

An important subset of PPTs are those which do not accept an empty subtrace, and so cannot include silent activities. We call these *loud trees* and denote the set of all such trees $\mathcal{PPT}_{loud}$.

$$\begin{aligned} \text{loud}\colon \mathcal{PPT} &\to \mathbb{B} \\ \text{loud}(x\colon w) &= (x \in A) \text{ where } x \in A \cup \{\tau\} \\ \text{loud}(\circlearrowright_p^\rho (u)\colon w) &= \text{ false} \\ \text{loud}(\circlearrowright_n^m (u)\colon w) &= \text{ loud}(u) \\ \text{loud}(\oplus(u_1, ..., u_m)\colon w) &= \bigwedge_i^m \text{loud}(u_i) \text{ where } \oplus \in \{\times, \wedge, \rightarrow\} \\ \mathcal{PPT}_{loud} &= \{u \in \mathcal{PPT} \mid \text{loud}(u)\} \end{aligned}$$

When every probabilistic loop operator $\circlearrowright_p$ in a tree is loud, the parent tree has the property of being *τ-bounded.* This means for some constant $k \in \mathbb{N}$, there are at most k silent transitions possible in any path through the tree.

5.3.2 Serial Trees

The concurrency operator $\wedge$ imposes no sequence constraint across child subtrees, beyond the constraints imposed by each subtree individually. The probability semantics are however dependent on the order of activities and silence in the possible paths. This makes the isolated calculation of subtree probabilities challenging. Serial trees are those parts of the tree not parented by a concurrent operator node ($\wedge$). Within serial trees, trace probabilities can be combined at the subtree level. The solution proceeds by cases of the different PPT operators, including some more fine-grained cases for loops.

Leaf Nodes Activity and silent nodes have a trace probability of zero or one.

$$\pi_{ppt}(\langle a\rangle, a\colon w) = 1$$
$$\pi_{ppt}(\langle\rangle, \tau : w) = 1$$
$$\text{otherwise, } \pi_{ppt}(\sigma, x\colon w) = 0 \text{ where } x \in A \cup \{\tau\}$$

Sequences and Fixed Loops Trace probability for sequences $\rightarrow$ first evaluates the empty trace probability. π_S then considers each possible non-empty two-way split of input trace σ. The function takes a trace, a sequence index, and a PPT model as parameters. This also applies to fixed loops $\circlearrowleft_n$. The definition makes use of the sequence slice operator $\ddagger$, defined in Section 2.1.1.

$$\begin{aligned}
&\pi_S\colon A^* \times \mathbb{N} \times \mathcal{PPT} \rightarrow [0,1], \text{ trace probability for sequences} \rightarrow\\
\pi_{ppt}(\sigma, \rightarrow(u_1, u_2, ...)\colon w) &= \pi_{ppt}(\langle\rangle, u_1) \cdot \pi_{ppt}(\sigma, \rightarrow(u_2, ...)\colon w)\\
&\quad + \pi_S(\sigma, 1, \rightarrow(u_1, u_2, ...)\colon w)\\
&\text{Function } \pi_S \text{ splits at index } n \text{ and recurses}\\
\pi_S(\sigma, n, \rightarrow(u_1, u_2, ...)\colon w) &= \pi_{ppt}(\sigma[1\ddagger n], u_1)) \cdot \pi_{ppt}(\sigma[n+1\ddagger|\sigma|], \rightarrow(u_2, u_3, ...)\colon w)\\
&\quad + \pi_S(\sigma, n+1, \rightarrow(u_1, u_2, ...)\colon w)\\
&\quad \text{where } n < |\sigma|\\
\pi_S(\sigma, |\sigma|, \rightarrow(u_1, u_2, ...)\colon w) &= \pi_{ppt}(\sigma, u_1) \cdot \pi_{ppt}(\langle\rangle, \rightarrow(u_2, u_3, ...)\colon w) \text{ , terminal case}\\
\text{otherwise, } \pi_S(\sigma, u) &= 0\\
\pi_{ppt}(\sigma, \circlearrowleft_n^m(u)\colon w) &= \pi(\sigma, \rightarrow(u, ...m \text{ times}...)\colon w)
\end{aligned}$$

Loud Sequences and Fixed Loops A simplification to π_S is possible for loud sequences and fixed loops, as the term considering the empty trace always evaluates to zero.

$$\pi_{ppt}(\sigma, \rightarrow(u_1, u_2, ...)\colon w) = \pi_S(\sigma, 1, \rightarrow(u_1, u_2, ...)\colon w) \text{ where } \operatorname{loud}(\rightarrow(u_1, u_2, ...)\colon w)$$

Choice Trace probability for the choice operator is a weighted sum of subtree probabilities.

$$\pi_{ppt}(\sigma, \times(x_1\colon w_1, ..., x_m\colon w_m)\colon w) = \frac{1}{w_T}\sum_i^m w_i \cdot \pi_{ppt}(\sigma, x_i\colon w_i)$$
$$\text{where } w_T = \sum_i^m w_i$$

Probabilistic Loops of Empty Traces The calculation for probabilistic loops $\circlearrowleft_p$ uses an approximation for the general case. However, there are some useful closed-form solutions for common special cases that we detail first. Empty traces have a closed form trace probability solution for probabilistic loops $\circlearrowleft_p$, even when silent leaf nodes are present. It uses the well-known sum of a geometric progression.

$$\begin{aligned}
\pi_{ppt}(\langle\rangle, \circlearrowleft_p^\rho (u)\colon w) &= \sum_{i=0}^{\infty} \frac{1}{\rho}(\frac{\rho-1}{\rho})^i \pi_{ppt}(\langle\rangle, u)^i \\
&= \frac{1}{\rho} + \frac{1}{\rho}\frac{1}{1-\frac{(\rho-1)}{\rho}\cdot \pi_{ppt}(\langle\rangle, u)} \quad \text{by sum of geometric series} \\
&= \frac{1}{\rho} + \frac{\pi_{ppt}(\langle\rangle, u)}{\rho^2}
\end{aligned}$$

Probabilistic Loops of Singletons Traces of one element (singleton traces) also have a solution, noting that for such traces, the loop must consume exactly one token on one iteration, among an arbitrary number of iterations consuming the empty trace.

$$\begin{aligned}
\pi_{ppt}(\langle a\rangle, \circlearrowleft_p^\rho (u)\colon w) &= \sum_{i=0}^{\infty} \frac{1}{\rho}\cdot \pi_{ppt}(\langle a\rangle, u)\left(\frac{\rho-1}{\rho}\right)^i \cdot \pi_{ppt}(\langle\rangle, u)^i \\
&= \pi_{ppt}(\langle a\rangle, u)\cdot \pi_{ppt}(\langle\rangle, \circlearrowleft_p^\rho (u)\colon w) \\
&\quad \text{Applying the result above for probabilistic loops for empty traces} \\
&= \frac{\pi_{ppt}(\langle a\rangle, u)(\rho-1)}{\rho^2} + \frac{\pi_{ppt}(\langle a\rangle, u)\cdot \pi_{ppt}(\langle\rangle, u)}{\rho^2}
\end{aligned}$$

Loud Probabilistic Loops In a loud loop, each iteration of the loop must consume at least one token in a given trace σ, meaning that iterations beyond $|\sigma|$ have probability 0.

$$\pi_{ppt}(\sigma, \circlearrowleft_p^\rho (u)\colon w) = \sum_{i=1}^{i=|\sigma|} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \text{ where } \operatorname{loud}(u)$$

Approximating Loops Including Silence In a probabilistic loop including arbitrary silent activities, the trace probability can be approximated within a probability bound ϵ. Loosely speaking, the loop is "unrolled" until the probability of any further execution is non-zero but very small.

Execution of a loop i times is the definition of a fixed loop, equivalent to a sequence $\rightarrow$ of length i

$$\begin{aligned}
\pi_{ppt}(\sigma, \circlearrowleft_p^\rho (u)\colon w) &= \sum_{i=0}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \\
&\text{Choose } k \text{ such that } \frac{(\rho-1)^{k+1}}{(\rho)^k} \leqslant \epsilon \\
\pi_{ppt}(\sigma, \circlearrowleft_p^\rho (u)\colon w) &= \sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \\
&\quad + \sum_{i=k+1}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft_n^i(u)\colon w) \text{ partition at } k \text{ iterations}
\end{aligned}$$

$$\sum_{i=k+1}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i = \sum_{i=0}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i - \sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \text{ expand term for iterations } > k$$

$$\text{Let } r = \frac{\rho - 1}{\rho} \text{ and note by definition } \rho > 0$$

$$\sum_{i=k+1}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i = r\frac{1}{1-r} - r\frac{1-r^{k+1}}{1-r} \text{ by sum of geometric series}$$

$$= \frac{r^{k+1}}{1-r} = \frac{(\rho-1)^{k+1}}{(\rho)^k}$$

$$\sum_{i=k+1}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \leqslant \epsilon \text{ by the definition of } k \text{ and } \epsilon$$

$$\text{Since probability bounds apply, } 0 \leqslant \pi_{ppt}(\sigma, \circlearrowleft^i_n(u)\colon w) \leqslant 1$$

$$\text{Replace } \sum_{i=k+1}^{i=\infty} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \text{ with } \epsilon \text{ to give the inequality}$$

$$\sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft^i_n(u)\colon w) \leqslant \pi_{ppt}(\sigma, \circlearrowleft^\rho_p(u)\colon w) \leqslant \sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_{ppt}(\sigma, \circlearrowleft^i_n(u)\colon w) + \epsilon$$

This final inequality puts upper and lower bounds on the π_{ppt} function, with a width of ϵ. Accordingly, an approximation function is

$$\pi_\epsilon(\sigma, \circlearrowleft^\rho_p(u)\colon w, \epsilon) = \sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^i \pi_\epsilon(\sigma, \circlearrowleft^i_n(u)\colon w, \epsilon)$$

The contribution of a subtree approximation range to a trace probability approximation never increases with inclusion in a larger tree. Consider PPT u with child u_s, and trace σ with subtrace σ_s. The trace probability for σ and u is conditional on the behaviour of u_s, or $\pi(\sigma, u) = Pr(\sigma \text{ matches } u | \sigma_s \text{ matches } u_s)$. As conditional probability is multiplicative by a value less than one, the probability π_ϵ and the approximation range ϵ are not increased, so the approximation range for u is bounded by ϵ.

5.3.3 Concurrent Trees

The entire subtree below a concurrent operator $\wedge$ is a *concurrent tree* (Definition 26). In concurrent trees, trace probabilities are calculated at the WSFA level. Above the topmost concurrent tree, the trace probability of the concurrent subtree can be used directly in other calculations, reducing their computational cost.

Concurrency in τ-bounded Trees Trace probabilities for concurrent trees depend on the WSFA trace probability defined in Section 5.2. For τ-bounded trees these are finite and the probability function will terminate.

$$\pi_{ppt}(\sigma, \wedge(u_1, u_2, ...)\colon w) = \pi_{wsfa}(\sigma, \mathbf{wa}(\wedge(u_1, u_2, ...)\colon w)$$

Concurrency in Probabilistic Loops A similar technique to serial trees can be used to generate approximated trace probabilities for probabilistic loops in concurrent trees. Given an ϵ bound,

$$\begin{tikzcd}
 & \rightarrow\colon 10 \arrow[dl] \arrow[dr] & & \\
\text{a}\colon 10 & & \times\colon 10 \arrow[dl] \arrow[d] \arrow[dr] & \\
 & \text{b}\colon 4 & \rightarrow\colon 4 \arrow[dl] \arrow[dr] & \tau\colon 2 \\
 & \tau\colon 4 & & \text{b}\colon 4
\end{tikzcd}$$

Figure 5.4: Example PPT u_{silent} with sequence, choice and a silent activity.

$$\begin{tikzcd}
 & \rightarrow\colon 10 \arrow[dl] \arrow[dr] & & \\
\text{a}\colon 10 & & \circlearrowright^{2}_{p}\colon 10 \arrow[d] & \\
 & & \times\colon 10 \arrow[dl] \arrow[dr] & \\
 & \text{b}\colon 8 & & \tau\colon 2
\end{tikzcd}$$

Figure 5.5: Example PPT u_{loop} with sequence, choice and probabilistic loop operators, as well as a silent activity.

choose k as for serial probabilistic loops, such that

$$\frac{(\rho-1)^{k+1}}{(\rho)^k} \leqslant \epsilon$$

An approximation function for loops is then, as above,

$$\pi_\epsilon(\sigma, \circlearrowright^{\rho}_{p}(u)\colon w, \epsilon) = \sum_{i=0}^{i=k} \frac{1}{\rho}\left(\frac{\rho-1}{\rho}\right)^{i} \pi_\epsilon(\sigma, \circlearrowright^{i}_{n}(u)\colon w, \epsilon)$$

As the contribution of subtree probabilities do not increase on inclusion in larger trees, the infinite set of paths generated by $\circlearrowright_p(u)$ may be truncated at k unrollings, as for serial trees. This allows a bounded probability approximation for all PPTs.

5.3.4 Example Trace Probability Calculation

In Figure 5.4 we have an example PPT with sequence, choice, and a silent activity. The PPT is a serial tree and τ-bound, so the probability can be calculated exactly.

$$\begin{aligned}
\pi_{ppt}(\langle a, b\rangle, u_{silent}) &= 1 \cdot \pi_{ppt}(\langle b\rangle, \times(b\colon 4, \rightarrow(\tau, b)\colon 4, \tau\colon 2)\colon 10) \\
&= \frac{4}{10} + \frac{4}{10} + 0 = \frac{4}{5}
\end{aligned}$$

In Figure 5.5 we have an example PPT with sequence, choice, a probabilistic loop, and a silent activity. The PPT contains a probabilistic loop including silence, so the probability must be approximated within a bound.

$$\pi_\epsilon(\langle a\rangle, u_{loop}, \frac{1}{10}) = 1 \cdot \pi_\epsilon(\langle\rangle, \circlearrowleft_p^2 (\times(b\colon 8, \tau\colon 2)\colon 10)\colon 10, \frac{1}{10})$$

$$\text{From parameters, } \rho = 2 \text{ and } \epsilon = \frac{1}{10} \text{ so } \frac{1^{k+1}}{2^k} \leqslant \frac{1}{10}$$

$$\text{Hence } k = 3$$

$$\begin{aligned}
\pi_\epsilon(\langle a\rangle, u_{loop}, \frac{1}{10}) &= \sum_{i=0}^{i=3} \frac{1}{2} \cdot \frac{1}{2^i} \pi_\epsilon(\langle\rangle, \circlearrowleft_n^i (\times(b\colon 8, \tau\colon 2)\colon 10), \frac{1}{10})\\
&= \frac{1}{2} \cdot 1 \cdot 1 \text{ for } i = 0\\
&\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{2}{10} \text{ for } i = 1\\
&\quad + \frac{1}{2} \cdot \frac{1}{8} \cdot \frac{2}{10} \cdot \frac{2}{10} \text{ for } i = 2\\
&\quad + \frac{1}{2} \cdot \frac{1}{16} \cdot \frac{2}{10} \cdot \frac{2}{10} \cdot \frac{2}{10} \text{ for } i = 3\\
&= 0.575 \text{ rounded to three places}
\end{aligned}$$

$$\pi_\epsilon(\langle a, b\rangle, u_{loop}, \frac{1}{10}) = 1 \cdot \pi_\epsilon(\langle b\rangle, \circlearrowleft_p^2 (\times(b\colon 8, \tau\colon 2)\colon 10)\colon 10)$$

$$k = 3 \text{ as above}$$

$$\begin{aligned}
\pi_\epsilon(\langle a, b\rangle, u_{loop}, \frac{1}{10}) &= \sum_{i=0}^{i=3} \frac{1}{2} \cdot \frac{1}{2^i} \pi_\epsilon(\langle b\rangle, \circlearrowleft_n^i (\times(b\colon 8, \tau\colon 2)\colon 10), \frac{1}{10})\\
&= 0 \text{ for } i = 0\\
&\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{5} \text{ for } i = 1\\
&\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{5} \cdot \frac{4}{5} \text{ for } i = 2\\
&\quad + \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{4}{5} \cdot \frac{4}{5} \cdot \frac{4}{5} \text{ for } i = 3\\
&= 0.312
\end{aligned}$$

The probability calculation for the concurrency operator is per the WSFA, and examples are found in section 5.2.

Using the example model in Figure 5.3, consider the probability that an approved claim is closed before the client is advised. The simplest scenario can be expressed as the trace $\sigma_{e1} = \langle \text{approve claim}, \text{close claim}, \text{advise claimant}\rangle$. Using the the formulae in this section, and an epsilon of 0.001, probability $\pi_{ppt}(\sigma_{e1}) = 0.021$, rounded to three places. Including cases where the claimant is advised up to five times, the probability is 0.040, or 4%.

5.3.5 Complexity

In evaluating computational complexity, we can recognise that the choices taken by the trace probability calculation form a tree of possible paths we will call a *path tree*. This is similar to a prefix tree [91, p492] or Petri net unfolding [69].

In the worst case, where all of the model is within a concurrent tree, the computational complexity of the path tree navigation is exponential, from the number of combinations of ordered strings. The bound for time is $O(n \cdot k \cdot 2^m)$ where n is the number of nodes in the original tree, k is the loop unrolling approximation bound, and m is the largest concurrent tree.

Let the size of the path tree model be n_t. The sequence and loop calculations consider $|\sigma|$ splits of input σ. The worst case complexity bound for the probability calculation is then $O(|\sigma|^2 \cdot n_t)$.

The path tree can be conceptual, or realised as a concrete data structure that can be reused on subsequent calculations, if the approximation bound is held constant. In this latter case, the memory complexity matches the time complexity for the worst case. This construction cost can then be treated as amortised over the lifetime of the data structure. One immediate use case is in conformance techniques where a stochastic language is constructed by calculating trace probability for every trace in an event log, such as Entropic Relevance [14].

Though the worst case bound is poor, the trace probability calculation is sensitive to model quality in both construction and trace calculation. Simple models reduce cost by reducing n_t. High fitness and low-precision models typically rely on loops for their generality, such as the Petri net flower model. The expansive state space will result in a large k and therefore n_t, where a more precise model will not. Determinism, sometimes identified as a quality measure in its own right [145], will also reduce the search space by reducing the branches needing to be searched for a particular trace. A balanced binary and deterministic path tree will have search cost $O(\log_2 n_t)$ for an overall cost of $O(n_t + |\sigma|^2 \log_2 n_t)$.

5.4 Results and Discussion

A prototype implementation of the trace probability calculation, using an in-memory path tree, is available in Haskell[1]. This includes unit tests.

Considering other solutions to trace probability, the linear programming approach [98] has a possible polynomial solution, rather than the worst-case exponential solution presented in this chapter. Implementations and extensions can also leverage the deep research literature on LP-solvers and optimised public implementations created for that problem. In this approach, there may be less opportunity to reuse calculation artifacts from one trace probability calculation to the next, as when calculating a stochastic language, even if using the same model.

Another solution to Trace-Prob uses probabilistic context-free grammars [157]. It uses a novel process model structure, the Probabilistic Generative Process Model (PGPM). Like the solution in this chapter, there is a relationship between the representation chosen for the calculation and process trees used elsewhere in process mining. PGPM representations are associated with grammar generation rules and estimated probabilities, and process trees are used as an input to that part of the algorithm. The PGPM work also discusses challenges arising from process tree concurrency operators. The PGPM method gives an approximate solution to trace probability, without approximation bound. An expectation minimisation (EM) algorithm is used, with each traversal having $O(|\sigma|^3)$ time complexity for trace σ.

[1]Source code for the implementation, experiment setup and result files are all available at `https://github.com/adamburkegh/toothpaste`. See Appendix A.

5.5 Summary

This chapter presented solutions for the trace probability problem, TRACE-PROB, on Probabilistic Process Trees (PPTs). An exact solution was provided for PPTs that produce paths with no more than k silent entries. A solution within a parameterised approximation bound was provided for the general case. This shows there are conformance use cases for PPTs, which were investigated from a discovery perspective in Chapter 4.

One avenue for future work is an experimental evaluation of trace probability solutions on real-life logs. Another is the extension of metrics for use on PPTs. A number of stochastic process quality metrics require stochastic languages. These metrics are often restricted to SDFAs, as they are a class of model with known techniques for generating these languages. One example is the Entropy Recall [101] measure. As deterministic PPTs are already equivalent to SDFAs, a particular virtue would be allowing the calculation of metrics on non-deterministic PPTs. It may also be possible to provide precise accuracy bounds for the metrics derived from the approximation bound for the trace probability calculation.

Both PPTs and the trace probability calculation on them are decomposable into recursive sub-components. This also suggests future work. Process mining techniques using models translatable to PPTs, such as many SLPNs, may leverage the techniques in this chapter. Recall that worst case complexity is sensitive to deep trees with concurrent parents, and when these are not present, solutions may be both computationally cheap and reused across traces. For example, a stochastic language calculation may be optimised by detecting a PPT-equivalent sub-component of a model, such as a τ-bounded tree. Alternatively a stochastic process discovery technique might internally calculate trace probability, to ensure that traces with high probability mass in the log are have a similar probability in the model.

Another solution to the trace probability problem is presented in Chapter 6, in the context of generating an approximated stochastic language from a process model. That solution treats SLPNs in general, but does not provide a guarantee relating the approximation parameter and the probability of a trace in the language.

Chapter 6

Stochastic Process Quality Dimensions

Geometry lies at the crossroads of a physics problem and an affair of the State.

Deleuze and Guattari
A Thousand Plateaus [58]

This chapter is about measuring and comparing stochastic process models, particularly those obtained through process mining [6].

Process mining has well-established ideas on the quality of models when they capture only a control-flow perspective, without a stochastic element. Many quantitative conformance measures for such models exist, organised under four quality dimensions: fitness, precision, simplicity and generalization [6, p118]. This approach supports thinking through design trade-offs in the construction of models, rather than seeking to optimise a singular metric.

In this chapter, we investigate *what dimensions may describe the quality of stochastic process models*. There are no established quality dimensions for such models. We use metrics designed for stochastic models as one starting point, as well as established control-flow process mining dimensions. The *mathematical* space under consideration is not purely analytical, as the underlying event logs to which models are compared are real-life empirical data on the social behaviour of organisations. This suggests using an exploratory quantitative analysis. Our approach was *empirical*, based on collecting and evaluating stochastic process models for real-life processes. Experiments generated a collection of thousands of models, using a variety of techniques, and based on event data from six real-life logs. Metrics collected included those from the literature and some adapted or designed specifically for this experiment. This empirical data was analysed for correlation and principal components (PCA). Based on this analysis, we propose three stochastic process model quality dimensions: *adhesion*, *relevance* and *simplicity*. Both the proposed dimensions for stochastic process model quality, and the empirical investigation, are novel.

The investigation includes:

- An experimental design for investigating stochastic process quality measure relationships;
- Metrics supporting that investigation;
- Two cycles of experimental evaluation and analysis;
- Candidate metrics based on the three dimensions; and
- Detailed demonstrations of the dimensions and metrics in use on concrete example models.

We also introduce the following secondary results:

- A practical, approximate solution for trace probability calculation (Trace-Prob, Definition 13);
- A genetic miner for the discovery of stochastic process models, Stochastic Evolutionary Tree Miner (SETM), suitable for laboratory use; and
- A new implementation of the entropic relevance measure [14] applicable to a broader range of models than those in the original paper or public implementation.

The remainder of this chapter proceeds as follows. Formal foundations are defined in Section 6.1 and background scholarship is discussed in Section 6.2. The experimental design is described in Section 6.3, including metric choice, model generation and differences between the first and second cycles of experiment. Results of the experiment are presented and analysed in Section 6.4. Quality dimensions, and metrics suggested by these results, are discussed in Section 6.5, including applying them to example models with a range of different qualities. Section 6.7 concludes.

6.1 Preliminaries

Extending the foundational structures in Chapter 2, we formalise the idea of generating a play-out event log from a stochastic model, and metrics on such logs.

Definition 36 (Play-out Log)**.** *A play-out log [6, p41] $L_p \in \mathcal{B}^+(A^*)$ is a finite real-valued multiset of traces.*

Real-life event logs have whole-numbered traces, but in our laboratory setup, fractional trace counts are useful in play-out logs to accommodate some side-effects of scaling. These are always positive. The set of all play-out logs is $\mathcal{L}^+ \supset \mathcal{L}$. Stochastic languages may be infinite, but the experiment design in this chapter uses finite approximations derived from *play-out logs.* The corresponding finite stochastic language for a play-out log can be found when scaling by the inverse of the cardinality of the log, $\frac{1}{|L|}$.

The concepts of metrics and measures from Definition 12 are also useful for play-out logs.

Definition 37 (Play-out Metrics and Measures)**.** *A* play-out metric *is a function comparing play-out logs and event logs, $m_p \colon \mathcal{L}^+ \times \mathcal{L}^+ \to \mathbb{R}$, and a* play-out measure *μ_p is a play-out metric with range* $[0, 1]$.

6.2 Related Work

This research builds on other scholarly work on stochastic process mining and models. This includes the discovery of such models, their quantitative measurement and comparison (conformance), and the dimensions along which they may be compared.

6.2.1 Discovery

Many control-flow discovery algorithms exist [72]. Stochastic process discovery algorithms are more limited in number, and may directly annotate models discovered by control-flow techniques [130] or construct stochastic models directly [117]. The current study builds directly on the analysis of genetically-mined control-flow models [33], both in study design, and direct extension of the Evolutionary Tree Miner code [34]. That work conducted a qualitative study on classes of models generated with different genetic miner constraints. In this work, the dimensions derived through quantitative analysis in Section 6.5 are applied qualitatively in Section 6.5.4. The Stochastic Evolutionary Tree Miner (SETM), a laboratory discovery technique suitable for exploring alternative models, is introduced in Section 6.3.3. For these two experiments, we used GSPN and SLPN discovery techniques with public implementations [130] that existed in the literature, including those in Chapters 3 and 4.

6.2.2 Quality Dimensions

For control flow process models and process mining, quality measures are typically considered to be measuring one of four quality dimensions: fitness, precision, simplicity and generalization [33], [6, p118]. Fitness measures indicate how well the model can reproduce the behaviour of the log. Precision measures how much of the model is used to reproduce log behaviour. A model may describe not just all traces in the log, but many other traces besides: such a model has high fitness but low precision. The simplicity dimension considers simpler models as higher quality, in both an application of Ockham's Razor [139] and a recognition that simpler models are easier to understand [115]. Generalization measures whether the model is applicable to more than the current sample (in process mining, a specific event log). In contrast to approaches in statistical learning where metrics aspire to represent overall model quality [153], in process mining, model quality is usually presented as a way to make design trade-offs against four quality criteria which are inherently in tension.

Though many studies investigate particular techniques quantitatively, quantitative experiments on the basis for quality dimensions are rarer. There is at least one quantitative study of the relationship between control-flow quality dimensions [87]. This used a collection of quality measures on models from a variety of control-flow discovery techniques. Factor analysis on the results found fitness and precision components with a clear correspondence to existing measures. An established consensus on what control-flow dimensions were meaningful preceded the experiment, and the empirical components supported those concepts. For the stochastic process context, there are no pre-known dimensions, so this study has a more exploratory character.

6.2.3 Conformance of Stochastic Process Models

Stochastic conformance metrics are those which specifically take stochastic process models as input. We make use of most of the metrics surveyed below in Section 6.3.1, either directly, or by introducing alternatives inspired by them (defined formally in 6.3.5). Existing metrics in the literature often consider probability mass or the probability of particular traces as parameters in metric calculation.

Calculating the probability of a particular trace through a process model is a non-trivial algorithmic problem, Trace-Prob [98] (Definition 13), and a solution for Probabilistic Process Trees is presented in Chapter 5, together with a discussion of existing work. In Section 6.3.2 we show an alternative solution which uses SLPN model play-out - a variant of the Petri net token game [6, p41] - to give the trace probability for all traces in a model beyond a given probability threshold. The result is represented as a play-out log.

The Earth-Movers' Distance measure [95] (EM) combines the well-known concepts of Levenshtein string edit cost - in this case in comparing traces - with the Earth-Movers' distance over the possible traces of model and log. As the set of possible model traces can be infinite, this includes a truncated measure using a specific fraction of the probability mass, for tractability.

Entropy has been used as one basis for model quality measurement. In projection-based precision and recall [101], Stochastic Deterministic Finite Automata (SDFAs) are constructed for both log and model. New SDFAs can be computed from a projection of the log over the model, and vice versa, and entropy ratios then provide measures for precision (H_P) and recall/fitness (H_F). These entropy precision and recall measures are used in this study. We also employ other measures inspired by them, but not limited to SDFAs (Play-out Entropy Fitness and Precision measures HIFT, HIPT, HJFT, HJPT). In entropic relevance [14], the process model is considered as a way of encoding the log. The entropy of the resulting encoding is then calculated using trace probability, and accounting for the encoding cost according to a background cost model. Three cost models are provided, with three corresponding metrics: Universal (HRU), Zero Order (HRZ) and Restricted Zero Order (HRR). In the original work trace probability is calculated for SDFAs only. In our experiments, we use a trace probability calculation from play-out logs when calculating these metrics, as seen in Section 6.3.1. The result is in bits and so not constrained to a $[0, 1]$ range.

The Alpha Precision measure [59] uses the stochastic language of the model and the event log, and inferences about the underlying system that generated the log. Model trace probability is aggregated for those traces where the probability of their occurrence in the underlying system exceeds a parameter called alpha significance. The resulting alpha significance parameter varies across domains, making the comparison of models across logs difficult. The Existential Precision metric (XPU) is the alternative we introduce to allow such comparisons.

In summary, current discovery techniques use a variety of techniques and output model types, but are somewhat comparable using a common denominator of SLPNs. Control-flow quality dimensions suggest starting points for the quality of stochastic models, but translation of the concepts to a stochastic setting is non-obvious, and new concepts may apply. Existing metrics for stochastic models are inconsistently related to control-flow dimensions, and have restrictions on supported model types due to the challenges of calculating stochastic languages. This landscape provides the challenges and constraints for our experimental design.

6.3 Experiment Design

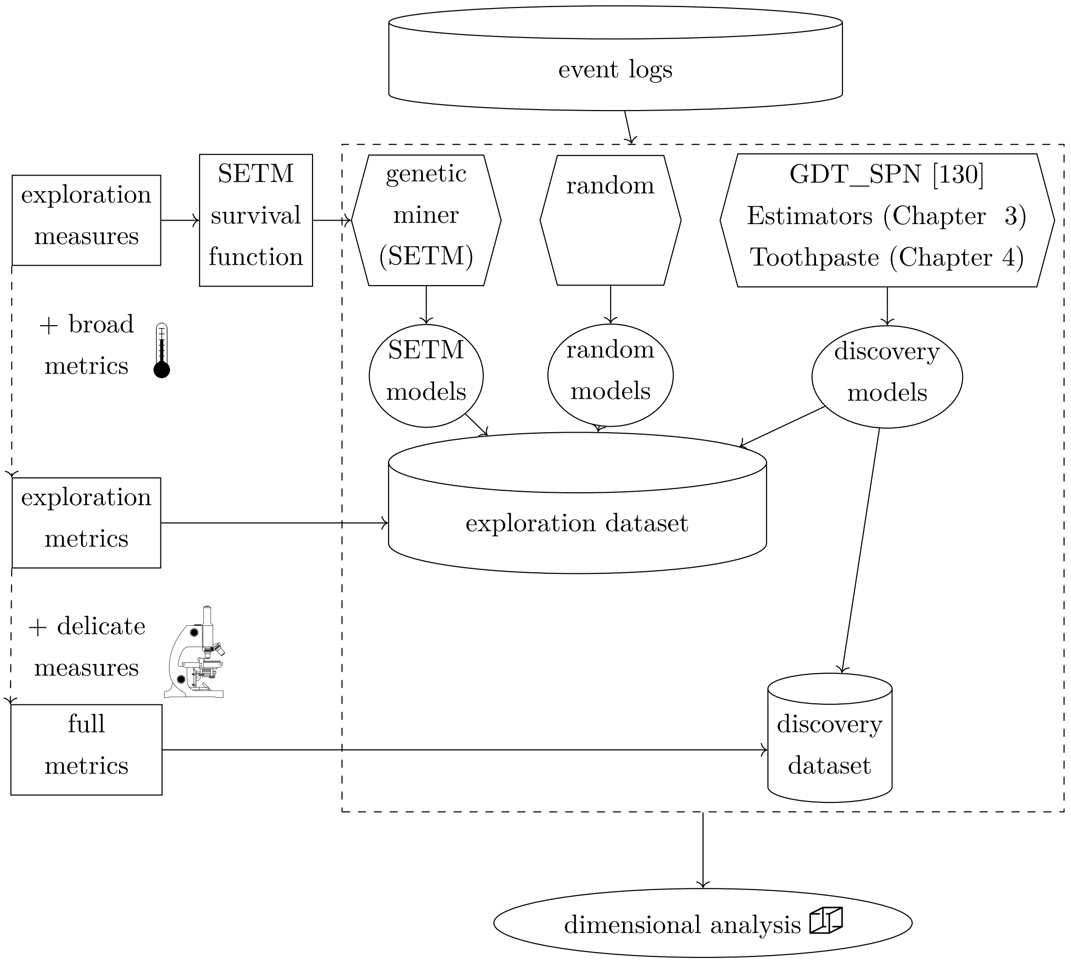


Figure 6.1: Experiment design, generating a broad range of models from event logs from different domains, then applying metrics to them for analysis.

Figure 6.1 shows the experiment design. Real-life event log data was used, covering a number of domains, as detailed in Tables 6.1 and 1.1. Using these inputs, a large number of process models were generated, using random generation and discovery techniques (including genetic miner SETM). This was designed to obtain a large number of models of varying quality, and an abundance of metrics for quantitative and qualitative analysis from different perspectives. Measures applicable to all models, including low-quality ones, are termed *exploration measures*. Some *broad metrics* applicable to all models (such as the number of edges) were added, and this set of *exploration metrics* were calculated for all models to form the *exploration dataset*. From laboratory experience, some measures in the literature only reliably return values on higher-quality models.

Table 6.1: Event logs for dimensions experiments..

Log	Traces	Variants	$\|A\|$	Domain
BPIC 2013 closed	1487	183	4	Issue tracking
BPIC 2013 incidents	7554	1511	3	Incident tracking
BPIC 2018 control	43808	59	7	EU Agriculture policy
BPIC 2018 reference	43802	515	6	EU Agriculture policy
Road Traffic Fines	150370	231	11	Italian policing
Sepsis	1054	846	16	Hospital diagnosis

We designate these *delicate measures*, and use them only on models generated by established discovery techniques, the *discovery models*. Though such models are only a smaller part of the larger universe of possible models, they are among the most relevant to process mining in practice. This full set of metrics was collected on the discovery models, yielding the *discovery dataset*. Two full iterations of the experiments were run, with small variations in the choice of metrics for each cycle. Finally, dimensional analysis was performed on both datasets.

In detailing the experiment design, we first introduce the metrics collected. Secondly, we examine the model representations which make the experiment practicable, stochastic play-out logs, and the technique used to construct them. Thirdly, we detail the model generation techniques used. Refinements and extensions from the first cycle of the experiments, to the second, are also discussed.

6.3.1 Choice of Metrics

As reviewed in Sections 6.2.3 and 6.2, metrics in the stochastic process mining literature are as yet limited in number, have restrictions on which models they can be applied to, and the result may not be comparable across models from different problem domains. In choosing and designing the metrics in these experiments, we deliberately cover a number of different design concepts. We drew on the four control flow quality dimensions, using existing stochastic metrics where possible (e.g., Entropy Precision [95]). To supplement these metrics and to explore a larger quality space, we constructed stochastic versions of control flow measures, such as Play-out Entropy Precision (HIPT) or the small changes to Generalization measures by trace floor and trace uniqueness [9]. We also explored the stochastic quality concepts Earth Movers' Distance, Probability Mass, and Entropy. The metrics used in the two experiment cycles are listed in Table 6.2. This includes the design concept behind a metric's inclusion, the abbreviation, and which experiments it was used for. Categories correspond to those in Figure 6.1. Formal definitions for metrics are found in 6.3.5.

Table 6.2: Metrics and their design rationale.

			Experiment	
Abbrv.	Metric Name	Design Concept	1	2
Exploration measures				
EMT	Earth Movers' With Play-out Trace	Earth Movers'	✓	✓
TOR	Trace Overlap Ratio	Probability Mass	✓	✓
TMO	Trace Probability Mass Overlap	Probability Mass	✓	
ARG	Activity Ratio Gower	Fitness	✓	✓
TRG2	Trace Ratio Gower length 2	Fitness	✓	✓
TRG3	Trace Ratio Gower length 3	Fitness	✓	✓
TRG4	Trace Ratio Gower length 4	Fitness	✓	✓
HIFT	Play-out Entropy Intersection Fitness	Fitness	✓	✓
HIPT	Play-out Entropy Intersection Precision	Precision	✓	✓
HJFT	Play-out Entropy Projection Fitness	Fitness	✓	✓
HJPT	Play-out Entropy Projection Precision	Precision	✓	✓
XPU	Existential Precision	Precision		✓
SSENC	Structural Simplicity by entity count [115]	Simplicity	✓	✓
SSEDC	Structural Simplicity by edge count [115]	Simplicity	✓	✓
SSS	Structural Simplicity incl. stochastic ratio	Simplicity	✓	✓
TGF1	Generalization by Trace Floor (1) [9]	Generalization	✓	
TGF5	Generalization by Trace Floor (5) [9]	Generalization	✓	✓
TGF10	Generalization by Trace Floor (10) [9]	Generalization	✓	
TGDU	Generalization by trace uniqueness [9]	Generalization	✓	✓
Exploration metrics				
CSS	Structural Complexity incl. stochastic	Simplicity		✓
MEC	Model Entity Count [115]	Simplicity		✓
MGC	Model Edge Count [115]	Simplicity		✓
HRU	Entropic Relevance w. Uniform [14]	Entropy		✓
HRZ	Entropic Relevance w. Zero Order [14]	Entropy		✓
HRR	Entropic Relevance w. Restricted Zero Order [14]	Entropy		✓
Delicate measures - discovery only				
EM	Earth Movers truncated 0.8 [95]	Earth Movers	✓	✓

Table 6.2 – continued from previous page

			Experiment	
Abbrv.	Metric Name	Design Concept	1	2
H_P	Entropy Precision [101]	Precision	✓	✓
H_F	Entropy Recall [101]	Fitness	✓	✓
Log metrics				
LTC	Log Trace Count	Log	✓	✓
LTE	Log Event Count	Log	✓	✓

In the first experiment cycle, we found that some of these exploratory measures were very highly correlated. As this is uninformative, and the measures were anyway excluded from much of the statistical analysis, Trace Overlap Ratio TOR, and two Generalisation by Trace Floor measures TGF1, and TGF10 were excluded in the second cycle of experiments. In the second experiment cycle, we were able to add new metrics based on new scholarly work, particularly Alpha Precision [59] and Entropic Relevance [14]. The Existential Precision (XPU) measure is based on Alpha Precision.

The Entropic Relevance [14] metrics are originally restricted to SDFAs in its description and public implementation. A new implementation for this work removes this restriction by using stochastic play-out logs.

Two log metrics were included as controls: Log Trace Count (LTC) and Log Event Count (LTE).

6.3.2 Stochastic Language Estimation with Play-out Logs

Play-out logs [6, p41] are an established process mining technique for generating event log traces based on process models. For place-transition Petri nets, a standard way of generating play-out logs is by "playing the token game" by noting the traces generated when the model advances from the initial marking through subsequent states. Play-out logs in a stochastic setting have an advantage over those with control-flow models: the stochastic model eliminates the need for arbitrary choices and assumptions when choosing between enabled transitions. Often, some assumed probability distribution is used for play-out logs on control-flow models Stochastic models, such as SLPNs, already include explicit probability functions which define behaviour when multiple transitions are enabled. The play-out log can then substitute for the model when comparing other logs or models, allowing measurement of models which otherwise could not be practically included in the experiment.

By using a finite representation to approximate the possibly infinite stochastic language of the model, a stochastic play-out log eliminates or greatly reduces the need for multiple samples to represent possible traces. Alternatives, such as random walks, will converge to representative values over many runs, and are necessary when a distribution is not well known, or when there are effects that emerge only after iterative calculation. The information in an SLPN allows alternative paths to be calculated in proportion directly when the goal is to obtain representative proportions of valid traces.

The stochastic play-out log generator implemented for these experiments is represented as function *spg*, which takes an SLPN and returns a play-out log. This can be thought of as a breadth-first search on possible traces, pruning improbable traces. To describe it, we overload function *eb* from Definition 5, which returns all enabled transitions for a net and a marking, to take SLPNs as input, and otherwise behave as for place-transition nets. Trace marking function $tg\colon \mathcal{SN} \times \mathcal{B}(P) \times T \to \mathcal{B}(P)$ returns the new marking after a transition fires. Lastly function *lab* gives a transition label as an activity sequence.

$$\begin{gathered} lab\colon \mathcal{SN} \times T \to A^* \\ lab((P,T,F,M_0,W,\lambda),t) = \langle\rangle \text{ if } \lambda(t) = \tau \text{ else } \langle\lambda(t)\rangle \\ \text{where } t \in T \end{gathered}$$

Definition 38 (Stochastic Play-out Generation)**.** *Let g be an SLPN model such that* $g = (P,T,F,M_0,W,\lambda)$. *Then sdlg is a play-out log generation functions taking an SLPN (g), a marking (m), a number of traces to be generated (b), and a maximum path length (ω). Function spg is a specialisation which assumes the initial marking. Function surplus allocates rounded amounts to specific traces.*

$$\begin{aligned} sdlg\colon \mathcal{SN} \times \mathcal{B}(P) \times \mathbb{N} \times \mathbb{N} &\to \mathcal{L} \\ sdlg(g,m,b,\omega) = \biguplus_{t \in eb(n,m)} &[\sigma^f \mid \sigma = lab(g,t) + \sigma_{tl} \\ &\wedge d = \text{floor}\left(\frac{bW(t)}{W_s}\right) + surplus(g,t,m,b) \\ &\wedge r = sdlg(g, tg(n,m,t), d, \omega - 1) \\ &\wedge \sigma_{tl} \in r \\ &\wedge f = r[\sigma_{tl}]] \\ \textit{where } W_s = \sum_{t' \in eb(g,m)} &W(t') \textit{ if } eb(g,m) \neq \varnothing \textit{ and } \omega > 0 \\ sdlg(g,m,b,\omega) = &[\langle\rangle^b] \textit{ if } eb(g,m) = \varnothing \vee \omega = 0 \\ spg(g,b,\omega_0) = &sdlg(g, M_0, b, \omega_0) \end{aligned}$$

The *spg* function takes a target size as a trace “budget”, then recursively splits the budget according to each possible state in a token game, and the relative weights of enabled transitions. The maximum path length ensures termination even for models that include potential livelock, or infinite loops. Traces affected by maximum path lengths are truncated.

Rounding is represented by the *surplus* function. The value rounded across all enabled transitions is the difference between the budget b and the sum of the weighted natural number allocations to those transitions. The rounding order is first to silent transitions, then by lexical order of the transition labels, then to the transition with the least allocation, then arbitrarily. Prioritising silent transitions favours the representation of loop exit states in the SLPN translation of PPT models. Least allocation refers to the enabled transition which will receive the least trace budget from the weighted allocation. This favours representation of rarer traces on the margin. In general, the design intent is for easily reproducible outputs, where variation is limited, and which discourages the computationally expensive process of sampling over multiple runs. Instead, if a given granularity is insufficient for a particular use, larger values for the log size and maximum path length parameters can be used to achieve more granularity.

As play-out logs can be straightforwardly converted to stochastic languages, this provides a practical approximation to the Trace-Prob problem for SLPNs from Definition 13. A play-out log M, generated without maximum path restrictions, will include all traces from paths which have a probability exceeding $\frac{1}{|M|}$. Some models have stochastic languages which fall outside this guarantee, when they have highly probable traces which exceed the maximum trace length. These models are rare in practice, and often amenable to inclusion by using a different maximum trace length. For example, long traces are mostly due to loop constructs in an underlying Petri net. In an SLPN which terminates, each iteration around the loop will construct new traces of monotonically decreasing probability. So very long traces produced by loops are often also very improbable.

The algorithm has a worst-case computational complexity of $O(b \cdot \omega_0)$, where b is the target log size, and ω_0 is the maximum path length. In many cases, a large maximum path length ($\omega_0 >> 0$) parameter is desirable to ensure representative and non-truncated traces. The use of the reachability graph (implicitly, via the token game) makes the algorithm combinatorial below this ceiling. The combination of the play-out log size limit and the maximum trace limit make it highly practical across a variety of models, as observed in this experimental work. Most of the exploration measures make use of this form of trace probability estimation, by using the frequency of traces in play-out logs. This approach both increased the range of possible models and radically decreased calculation times.

In the implementation, the play-out log size was set to 1000 traces. The maximum path length was set to 5000 for the first round of experiments and 500 in the second. Almost all play-out logs experiencing maximum path truncation were from random models, and on a minority of traces.

6.3.3 Stochastic Evolutionary Tree Miner (SETM)

Genetic algorithms try a broad range of solutions according to a process loosely inspired by the "survival of the fittest" genetic adaptation of biological species to their environment. These algorithms usually start with some randomly generated potential solutions. The solutions are evaluated according to a survival function, and the best kept. These are then mutated randomly according to set rules, and the process is repeated for many iterations, or *generations*. When employed for process discovery, these algorithms are termed genetic miners [34].

A novel genetic miner for discovering stochastic process models, the Stochastic Evolutionary Tree Miner (SETM), was implemented for these experiments. It is based on the Evolutionary Tree Miner [34]. The SETM generates random PPTs for the initial generation of models. Four possible mutations are then applied: to add a node (including control flow nodes and silent transitions), mutate a single node, remove a subtree, or remove useless nodes (specifically to apply Preserving Compression rules, detailed in Chapter 4). These mutations preserve valid and consistent tree weights. Models were exported as SLPNs. SETM is suitable for exploring model alternatives in a laboratory setting, with the quality of final generation models being far higher than random models, but not at the same level as those from other discovery techniques. An example mutation is shown in Figure 6.2.

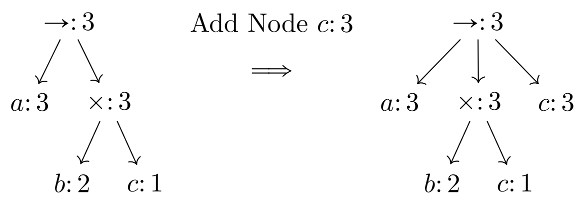


Figure 6.2: An example of applying an Add Node mutation on a PPT. The activity to add and the location in the tree are chosen randomly during mutation.

The SETM was run across 1000 generations with a survival function incorporating all the exploration measures for that cycle, with equal weight. The model with the highest survival score in each generation was added to the exploration dataset, generating a spectrum of models of moderate quality. Any additional exploration metrics were also collected for each model. The genetic miner yielded results for four of the logs in this experiment; due to timeouts after forty hours, the two logs with the most activities gave partial results which were not included.

6.3.4 Model Generation

As well as genetic mining, the two other classes of model generation techniques employed were firstly, random generation, and secondly, existing stochastic discovery techniques.

Random models were created by randomly choosing nodes of *Probabilistic Process Trees (PPTs)* (see Definition 25). The random generation included silent, activity and control flow nodes. Models larger than the arbitrary cutoffs of a tree depth of 30 or 1000 transitions were discarded, and substituted for another generated model. Models generated randomly were anticipated to have lower quality.

Models generated by existing stochastic discovery techniques were also included, and were anticipated to be of higher quality. Public implementations of stochastic process discovery techniques for GSPNs (those in chapters 3 and 4, and GDT_SPN discovery [130]) created a further 103 models relating to the selected event logs. State of the art stochastic discovery techniques yield higher quality models than the other two generation methods, but still yield low-quality models in a number of cases. This meant the discovery dataset still contained a wide range of metric values.

A total of 9301 models were generated. Metric implementations, SETM, metric reuse, and other experimental scaffolding, was all implemented in Java using the ProM framework[1]. Experiments were run on a Linux clustered data centre using 50 Gb of RAM.

6.3.5 Detailed Exploration Metrics

This section details the measures summarized in Table 6.2. For the measure definitions below, let event log $L \in \mathcal{L}$, model $g \in \mathcal{SN}$, $\omega_0 \in \mathbb{N}$. To obtain the play-out log $M \in \mathcal{L}^+$, the model g is played out to k traces, then occurrences are scaled to match the original log: $M = \frac{|L|}{k} \cdot spg(g, k, \omega_0)$.

[1]. All source code is accessible at `https://github.com/adamburkegh/spm_dim`. See Appendix A.

The first measure is a simplification of the stochastic Earth Movers' distance [95].

EMT Earth Movers with play-out trace weighting.

$$EMT(M, L) = 1 - \frac{1}{|L|} \sum_{\sigma \in L} \max(L[\sigma] - M[\sigma], 0)$$

Two measures address how much of the probability mass of the log is in shared traces.

TMO Trace Probability mass overlap.

$$TMO(M, L) = \sum_{\sigma \in L \mathbin{\cap\!\!\!\!\!{\scriptscriptstyle +}} M} \frac{(L \mathbin{\cap\!\!\!\!\!{\scriptscriptstyle +}} M)[\sigma]}{|L|}$$

TOR Trace overlap ratio.

$$TOR(M, L) = \frac{|L \mathbin{\cap\!\!\!\!\!{\scriptscriptstyle +}} M|}{|L|}$$

Analysis of which subtraces occur in both log and model (represented by the play-out log) approximate fitness.

ARG The Gower's similarity [75] between activity count ratio vectors. This measure is designed to be deliberately sensitive to variation between poor quality models, when other measures may be zero. Given log L, take $ST_n(L)$ to be the subtraces of length n, $\sigma_s \# L$ the subtrace frequency of σ_s, with each occurrence in a trace counted, and $||L||_n$ to be the total subtraces of length n. ARG is a special case: ARG=TRG1.

TRGn Subtrace ratios, activity ratios generalized to sub-traces of length n. **TRG2**, **TRG3** and **TRG4** are all measured.

$$TRGn(M, L) = \sum_{\sigma \in ST_n(L \uplus M)} 1 - y_\sigma$$

$$\text{where } y_\sigma = \frac{1}{\max(\sigma \# L, \sigma \# M)} \left| \frac{\sigma \# L}{||L||_n} - \frac{\sigma \# M}{||M||_n} \right|$$

Two simplified variants of evaluation measure entropy [100], based on play-out logs, are used to define fitness and precision measures. The first uses bag intersection.

HIFT Play-out entropy intersection fitness.

$$HIFT(M, L) = \min(1, \frac{H(L \mathbin{\cap\!\!\!\!\!{\scriptscriptstyle +}} M)}{H(L)})$$

HIPT Play-out entropy intersection precision.

$$HIPT(M, L) = \min(1, \frac{H(L \mathbin{\cap\!\!\!\!\!{\scriptscriptstyle +}} M)}{H(M)})$$

The second entropy variant uses SDFA projection [100] function $\mathcal{P} \colon \mathcal{L}^+ \times \mathcal{L}^+ \to \mathcal{L}^+$, where traces are used as SDFA tokens.

$$\mathcal{P}(L_1, L_2) = L_P \uplus [\langle\rangle^{|L_1| - |L_P|}]$$

$$\text{where } L_P = [\sigma^i \in L_1 \mid \exists_{j>0}\, \sigma^j \in L_2]$$

HJFT Play-out entropy projection fitness.

$$HJFT(M, L) = \frac{H(\mathcal{P}(L, M))}{H(L)}$$

HJPT Play-out entropy projection precision.

$$HJPT(M, L) = \frac{H(\mathcal{P}(M, L))}{H(M)}$$

XPU Existential precision adapts Alpha precision [59] by calculating the probability mass of model traces represented at least once in the log.

$$XPU(M, L) = \frac{1}{|M|} \sum_{\sigma in L} M[\sigma]$$

Three simplicity measures are scaled by log size to impose a valid upper bound of 1.

SSENC Structural simplicity by entity count [114].

$$SSENC(g, L) = \max(1 - \frac{|P| + |T|}{|L|}, 0)$$

SSEDC Structural simplicity by edge count [114].

$$SSEDC(g, L) = \max(1 - \frac{|F|}{|L|}, 0)$$

SSS Structural simplicity by all structural components in SLPNs. This accounts for stochastic features not found in existing structural simplicity measures.

$$SSS(g, L) = \max(1 - \frac{1}{|L|}(|P| + |T| + |F| + |\bigcup_{t \in T} W(t)|), 0)$$

The following generalization measures are at a trace level, and are taken from example measures in [9].

TGF1 Generalization by trace floor, gen_{L2M_q} [9]. We also use **TGF5** and **TGF10** as measures for trace floors of 5 and 10 respectively.

$$TGF1(M, L) = \frac{|[\sigma \in l | \sigma \in M \wedge L[\sigma] \geqslant q]|}{|L|} \text{ with } q \geqslant 1$$

TGDU Generalization by trace uniqueness difference, $gen_{L2M_{HB}}$ [9].

$$TGDU(M, L) = \frac{|[\sigma \in L | \sigma \in M]| - |L \Cap M|}{|L|}$$

CSS Structural Complexity incl. stochastic includes both control flow and stochastic features of a SLPN in a common metric. It is a denormalised inverse of SSS.

$$CSS(g, L) = |P| + |T| + |F| + |\bigcup_{t \in T} W(t)|$$

MEC Model entity count is a count of places and transitions.

$$MEC(g, L) = |P| + |T|$$

MEC Model edge count is a count of connecting arcs.

$$MEC(g, L) = |F|$$

6.4 Results

An exploratory quantitative analysis was performed on model measures from Experiments 1 and 2.

6.4.1 Quantitative Analysis For Component Identification

Analyses of correlations and principal components [88] were performed to determine commonality and orthogonality between metrics, that indicated potential quality dimensions. To weigh the sources of models equally, sources with less than 1000 models had data points repeated as if resampled. Sample sizes are quoted without resampling. Scaled PCA was used, centring all input parameters to a zero mean and scaling to unit variance. This allowed metrics such as HRU to be included in the analysis, even though they could not be included in the genetic miner survival function, as their range was not known in advance.

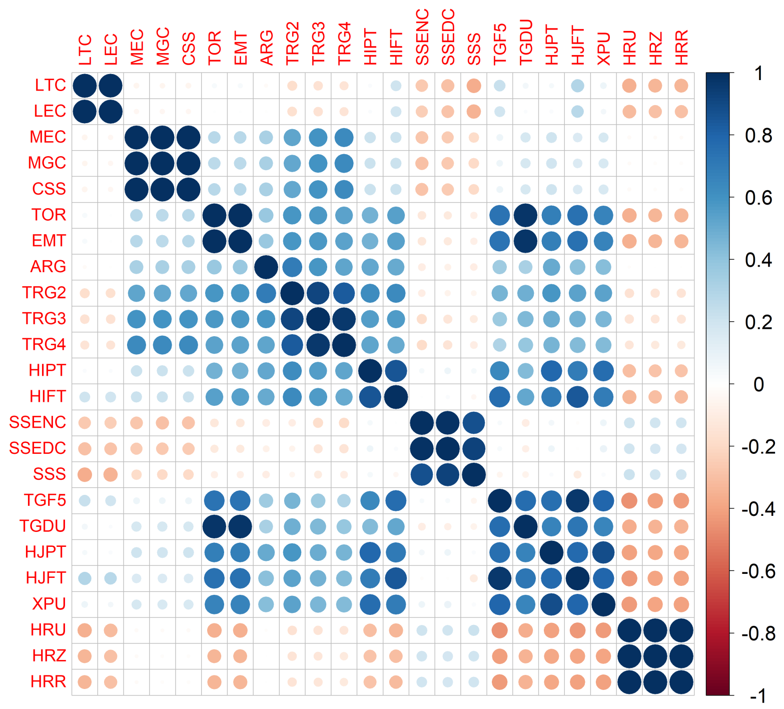


Figure 6.3: Correlation between exploration metrics, Experiment 2.

In Experiment 1, some measures were very highly correlated (> 0.99), and these were excluded in Experiment 2, as indicated in Table 6.2. Some of these measures may, in retrospect, be theoretically equivalent. Experiment 2 metric correlation was examined for the exploration and discovery datasets; exploration metrics are shown in Figure 6.3. Correlation is indicated in blue and anti-correlation in red, with colour intensity and circle size indicating the strength of correlation. A number of correlated groups of measures can be observed in these results, and the metrics are ordered so they are clearer visually. A number of groups are already related by concept and implementation: metrics for logs (LTC,LTE), model complexity (MEC,MGC,CSS), Trace Ratios Gower (TRG2-4), simplicity (SSENC,SSEDC,SSS), and Entropic Relevance (HRZ,HRR,HRU). Some metrics showed high correlations even though they were included under different concepts. While Trace Overlap Ratio (TOR) and Earth Movers' With Play-out Trace (EMT) are included under Probability Mass and Earth Movers' respectively, the Earth Movers' Distance measure definition is closely related to probability mass. Play-out Entropy Intersection Fitness and Precision (HIFT,HIPT) are correlated, though they are intended to measure quite distinct control-flow concepts. These measures do share implementation similarities, in that they both use an entropy calculation over a trace projection. More surprising, perhaps, is the group of five partially correlated metrics TRG5-XPU, which includes metrics intended to measure fitness, precision and generalisation, all together.

It is also interesting to note which metrics are not correlated or are anti-correlated. Activity Ratio Gower (ARG) is not strongly correlated with any other metric, including other subtrace ratios (TRG2-4). Metrics for fitness are not strongly correlated with one another, and similarly for precision and generalisation. The Entropic Relevance metrics (HRZ,HRR,HRU) show some anti-correlation with the TRG5-XPU grouping, and with some other tracewise metrics.

The two log metrics LTC and LTE showed correlations with the trace ratio measures and the simplicity measures (-0.45 and -0.63 respectively). In these cases, either the number of traces or events is a parameter to the measure, so this is to be expected. Correlation between LTC and LTE and other metrics is low. As these properties were already known, the two log metrics were then excluded. An Anderson-Darling test for normality showed no variables fit a normal distribution ($p < 0.001$), ruling out techniques such as factor analysis for both experiment cycles.

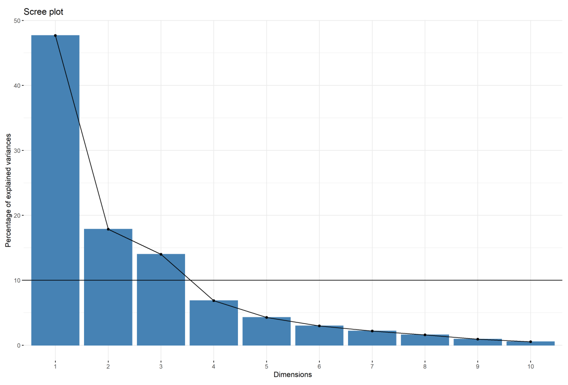


Figure 6.4: Scree plot of percent of variance explained by each principal component, sorted in descending order, on PCA for exploration metrics in Experiment 2.

A scaled Principal Component Analysis (PCA) was then used to examine the basis for orthogonal components. PCA outputs a change of basis for a dataset with n measures in which the resulting n dimensions can be ranked by their maximisation of variance. The result is guaranteed to produce orthogonal dimensions (in PCA terminology, components), and is often used for dimensional reduction by choosing the highest-ranked components. It is employed here to identify potential orthogonal dimensions with an empirical basis. A scree plot of the variance covered by the components was used to estimate the number of possible dimensions. Figure 6.4 shows the scree plot for exploration metrics in Experiment 2. These three components explain 39.3%, 27.5% and 11.3% of the variance respectively, and the remaining components explain at most 6% each. Results for the discovery dataset, which includes delicate measures, were similar, explaining 45.9%,

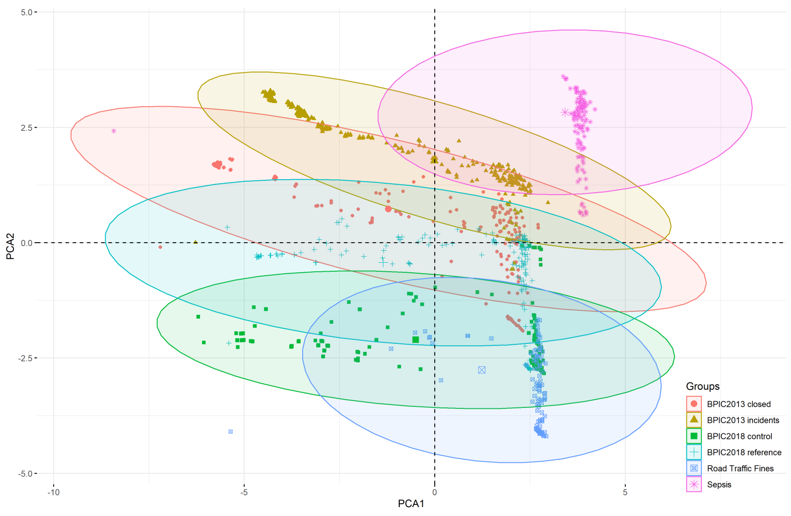


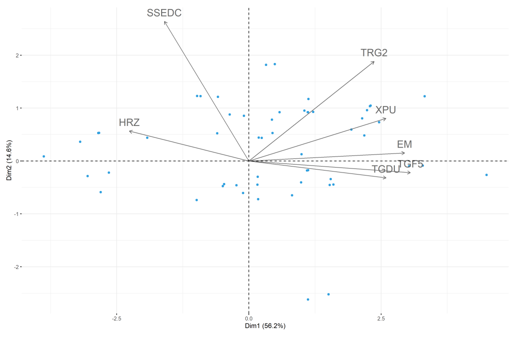
Figure 6.5: Exploration dataset scatterplot against PCA components 1 and 2. Ellipses and colour distinguish source logs.

16.9% and 12.3% respectively; these are also similar to Experiment 1's results. Both experiments showed three orthogonal components, with a possible fourth, using the elbow technique. This fourth component was not clearly identified with an underlying concept, and explained less than 10% of the variance. We chose to not include it, for a more conservative analysis, and as it was not identified with an underlying concept.

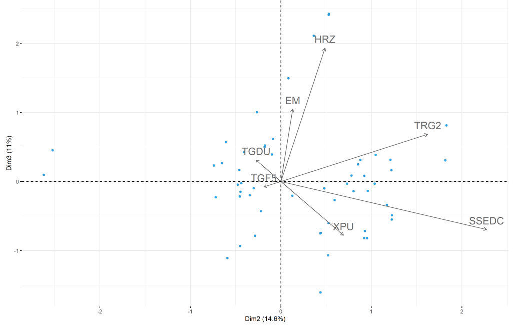
We performed robustness tests to examine whether components could be identified with any element in the experiment setup itself, and for consistency across data subsets. Specifically, components were compared to log sources and to model sources (i.e. random/SETM/discovery) to check whether the PCA was simply identifying these input partitions. Classification by log and by model source varied across PCA components for both experiments. Figure 6.5 shows one example classification by log across the first two PCA components for the exploration dataset. Though models from different logs, as represented by the ellipses, have different quality profiles, they are not straightforwardly identified with components in either instance. Since this analysis suggests the components reflect deeper underlying regularities in the dataset, a second round of analysis, below, breaks these components down further.

6.4.2 New Metrics Yield New Components

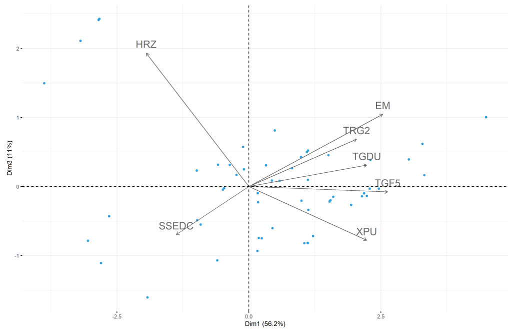
Different components were identified by the two cycles of experiments. Within experiments, components differ across exploration and discovery datasets, and the order of influence of the second and third components changes, but similar metrics were associated with them in both cases. In both experiments, the first component is associated with the Earth Movers (EM) and Trace Generalization by floor (TGF) measures. A second component is associated with Simplicity measures. However, in Experiment 1, the third component has an association with Entropy Precision and Recall (H_P,H_F) and Trace Ratio measures. In Experiment 2, the Structure Stochastic Complexity



(a) First versus second PCA components.



(b) Second versus third PCA components.



(c) First versus third PCA components.

Figure 6.6: PCA biplots for selected metrics on the discovery dataset, comparing three components. The approximate orthogonality of the Native Metrics Earth Movers' Distance (EM), Simplicity by Edge Count (SSEDC), and Entropic Relevance Zero Order (HRZ) can be observed.

(CSS) and on the discovery dataset, the Entropy Relevance metrics (HR*), are closely associated with a third component, and not correlated with Entropy Precision and Recall.

To clarify these relationships and seek a more parsimonious description of the data with fewer input metrics, we performed a second round of analysis. Metrics that correlated with another at > 0.9 were pruned. When choosing from a pair of correlated metrics, we prioritised first metrics from published literature, then metrics that correlated to delicate measures on the discovery dataset, then conceptually clearer metrics. By conceptual clarity, we refer to a decision about whether to include the complexity metric CSS, or the simplicity metric used as an input to its calculation. Since CSS has a known formal relationship to simplicity measure SSS, one of these metrics could be excluded. Examining PCA biplots, the resulting dimensions were more clearly aligned with named measures, and hence existing concepts, when based on simplicity, so this was the metric included. PCA biplots plot measures against two selected PCA components, which Figure 6.6 illustrates for the discovery dataset in Experiment 2. The six metrics, and the PCA performed on them across both Experiment 2 datasets, are the immediate underlying data for our proposed quality dimensions.

In summary, three PCA components are shown across both experiments and across exploration and discovery datasets. Two components are similar across the two experiments; a third differs in composition under the influence of new metrics. A second analysis, centred on the Experiment 2 metrics, suggests candidates for the quality dimensions proposed in Section 6.5.

6.5 Quality Dimensions

From the experimental results above, we propose three quality dimensions, which we name *Adhesion, Simplicity, and Relevance*, and which we characterise below. To apply these dimensions for quantitative measurement and comparison, we provide three sets of measures, corresponding to two interpretations of the experiments. In the first view, the experiments and analysis are considered to have revealed hidden underlying regularities, akin to physical laws, and corresponding to PCA components, which the combined weighted measures approximate. We call this view *dimensional realism.* In the second view, the experiments and analysis are used to reveal which metrics effectively partition the quality space, by capturing variance and their orthogonality to other metrics. Those metrics are then used directly, using only scaling, and so this is termed the *native metrics* view. Though the problem of choosing synthetic or direct metrics is not a new one in science, the dimensional realist / native metrics terminology is, to our knowledge, new, at least as applied to the specific problem of dimensional choice.

6.5.1 Three Model-Log Quality Dimensions

Adhesion To represent how little effort is required to transform one stochastic language into another, we use the term *adhesion.* Such a transformation can involve both modifying which traces the process accepts, and the probability of those traces. An informal interpretation is how few changes a team needs to make to adhere to a different way of working.

Relevance Relevance measures the informational cost of reconstructing the complete traces from the event log with the model. The dimension name is directly inspired by the Entropic Relevance metrics [14][2]. This definition also constrains the concept to a *trace level* view of the model and log, where completed cases are considered relevant, but even slightly differing traces are not.

Simplicity Parsimony in models is well-recognized as a virtue in science generally [139], and a desirable dimension in process models specifically [6, p118]. Simplicity represents the number of explicit syntactic features of the model. Encoding costs, or the movement of complexity into a notation, are not considered, but are mitigated in this work by using SLPNs as a common model structure. Syntactic simplicity, as in this dimension, is also distinct from behavioural simplicity [89], for example where a model with many elements might describe a process with very limited behaviour. Behavioural simplicity is more associated with the Relevance dimension.

[2]As Relevance concerns the amount of information shared between log and model, it is related conceptually to Entropy, the name used for the second dimension in an earlier conference paper [36]. However, as noted in Section 6.4, the underlying metrics differ significantly.

Example models illustrating these dimensions are examined in Section 6.5.4 and in Figures 6.8 and 6.9.

6.5.2 Dimensional Realist View

In the dimensional realist view, PCA components are taken as the model of the underlying space. After excluding highly correlated metrics, we use those remaining as a synthetic estimator for each dimension.

In our Principal Component Analysis, each element is centred by its mean and scaled by its standard deviation. Take the metrics $m_1 ... m_6$ included in the analysis, then m_i to be the i-th metric, $\bar{x_i}, s_i$ to be the corresponding mean and standard deviation, and P_{Xi} the PCA loading for one of the PCA components.

$$
\begin{aligned}
X_D &= P_{X1}\frac{m_1 - \bar{x_1}}{s_1} + P_{X2}\frac{m_2 - \bar{x_1}}{s_2} + ... \\
&= \frac{P_{X1} \cdot m_1}{s_1} - \frac{P_{X1} \cdot \bar{x_1}}{s_1} + \frac{P_{X2} \cdot m_2}{s_2} - \frac{P_{X2} \cdot \bar{x_2}}{s_2} + ... \\
&= \frac{P_{X1} \cdot m_1}{s_1} + \frac{P_{X2} \cdot m_2}{s_2} + ... - \sum_{i=1...6} \frac{P_i \bar{x_i}}{s_i}
\end{aligned}
$$

This linear equation is reorganised by noting that the concluding sum is a constant, and renaming the constant factors $\frac{P_{Xi}}{s_i}$ after their corresponding metrics.

Definition 39 (Dimensional Realist Metrics)**.**

$$
\begin{aligned}
X_D = {} & MX_{XPU} \cdot XPU + MX_{TGF5} \cdot TGF5 + MX_{TGDU} \cdot TGDU \\
& + MX_{HRZ} \cdot HRZ + MX_{TRG2} \cdot TRG2 + MX_{SSEDC} \cdot SSEDC + K_{XD} \\
\textit{where } K_{XD} = {} & \sum_{i=1...6} \frac{P_{Xi} \cdot \bar{x_i}}{s_i} \\
& \textit{and } X \in \{A, R, S\} \textit{ for (A)dhesion } A_D\textit{, (R)elevance } R_D \\
& \textit{and (S)implicity } S_D\textit{, respectively.}
\end{aligned}
$$

The empirically derived factors and constants are summarised in Table 6.4. Min/max scaling has been applied. For the third dimension, simplicity, the P_X values for the entropic relevance (HRZ) factor is zero at four digits of significance, excluding it. The values for existential precision XPU are very small (-0.00113), such that it could also be excluded. The PCA component for simplicity minimised when input simplicity was highest, so we have reversed the signs of the factors and constant so that high values indicate higher quality. The factors show the conceptual limitations of dimensional realism: the explicit simplicity measure based on edge count SSEDC has a similar contribution to both Relevance and Simplicity DR dimensions, and the relevance metric HRR contributes only slightly more to Relevance than Adhesion.

Unscaled factors, constants and min/max values for this study are in Tables 6.4 and 6.5. The min/max scaled values in Table 6.3 are derived directly from these.

Table 6.3: Dimensional realist factors and constants from exploration metrics, after min/max scaling.

Metric	Adhesion (MA)	Relevance (MR)	Simplicity (MS)
XPU	0.168	-0.0675	-0.00113
TGF5	0.170	0.0349	-0.0916
TGDU	0.158	0.05637	0.109
HRZ	-0.00526	-0.00479	0.000
TRG2	0.121	-0.344	-0.463
SSEDC	0.0674	-0.210	0.325
Constant	Adhesion (K_{AD})	Relevance (K_{RD})	Simplicity (K_{SD})
	0.342	-0.285	0.126

Table 6.4: Dimensional realist factors and constants from exploration metrics, no scaling.

Metric	Adhesion (MA)	Relevance (MR)	Simplicity (MS)
XPU	1.518	-0.693	-0.087
TGF5	1.540	0.358	-0.709
TGDU	1.425	0.579	0.842
HRZ	-0.0476	-0.0492	0.000
TRG2	1.090	-3.527	-3.582
SSEDC	0.610	-2.157	2.518
Constant	Adhesion (K_{AD})	Relevance (K_{RD})	Simplicity (K_{SD})
	3.09	-2.92	0.97

The calculations in Definition 39 yield metrics rather than measures, as they may range beyond $[0, 1]$, including negative values. Min/max scaling was applied to achieve a measure in an applied setting, as seen in Table 6.3. They are derived by noting that five of the six input metrics are already normalised within a $[0, 1]$ range. The remaining input metric, Entropic Relevance Zero Order (HRZ), is observed in experimental data to have a maximum of 52.55 and range of 50.68. To calculate scale ranges, we treated HRZ as having range $[0, 60]$, and calculated overall ranges based on the theoretical extremes of each input variable.

6.5.3 Native Metrics View

In the native metrics view, a small number of metrics that partition the space are each identified with a particular dimension. The metrics are not fully orthogonal, in the sense that they show partial correlation and do not intersect at perfect right angles. Compared with the dimensional realist view, this loses some information from the excluded metrics, but it is simpler. Since metrics were originally designed with the intent to capture some particular aspect of model quality, it is also conceptually clearer. Native metrics are listed in Table 6.6. Figure 6.7 plots a PCA using only the selected metrics on the exploration dataset. As the current implementation of Earth Movers'

Table 6.5: Dimensional realist min/max from exploration metrics.

	Adhesion A_{DM}	Relevance R_{DM}	Simplicity S_{DM}
Minimum	-5.94	-6.40	-5.36
Maximum	3.09	3.87	2.38

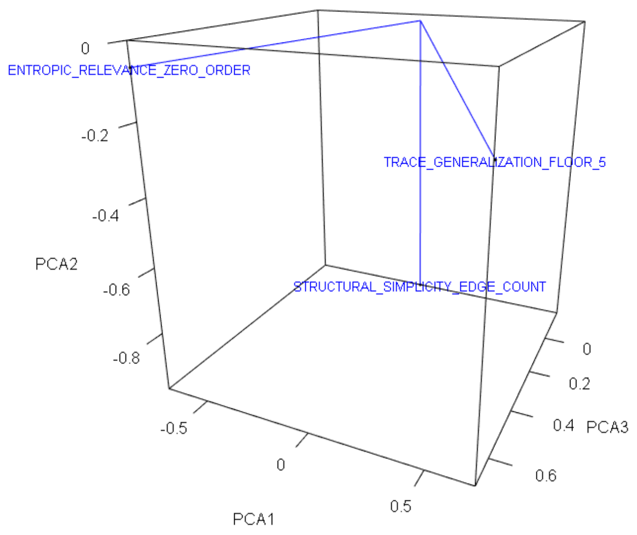


Figure 6.7: 3D plot of metrics Trace Generalization by Uniqueness (5) (TGF), Entropic Relevance Zero Order (HRZ), and Simplicity by Edge Count (SSEDC), against PCA dimensions for those three metrics, on the exploration dataset.

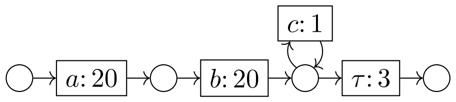
Distance is a delicate measure, we include a substitute of TGF5 (correlation = 0.75) for when it is unavailable.

Table 6.6: Native metrics for Adhesion, Relevance and Simplicity, chosen by joint PCA orthogonality and conceptual linkage.

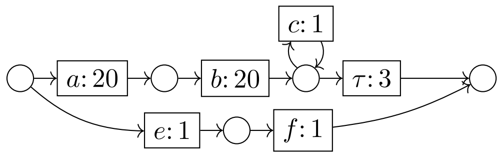
Data Set	Adhesion	Relevance	Simplicity
Exploration	Generalization by Trace Floor (5) TGF5	Entropic Relevance w. Zero Order HRZ	Structural Simplicity by edge count SSEDC
Discovery	Earth Movers truncated EM	Entropic Relevance w. Zero Order HRZ	Structural Simplicity by edge count SSEDC

6.5.4 Dimensions In Use On Example Models

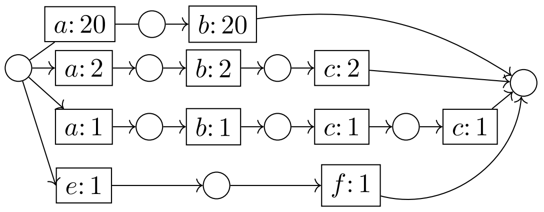
To give a sense of the three dimensions, and metrics that approximate them, Figures 6.8 and 6.9 illustrate extreme cases. These can serve as paradigmatic examples of the dimensions. Cases were informed by using alternative fitness functions for SETM which neglected one or more dimensions,



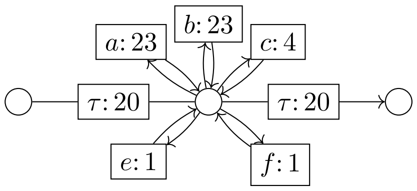
(a) Adhesion+ Relevance+ Simplicity+. Covers the bulk of the probability mass and the completed traces in the log.



(b) Adhesion++ Relevance++ Simplicity mid-range. Near-perfect probability mass and completed traces, as well as fitness and precision.



(c) Adhesion++ Relevance++ Simplicity-. Trace Model.



(d) Adhesion- Relevance- Simplicity mid-range. Flower Model with event frequencies.

Figure 6.8: Models exemplifying adhesion and entropy variations relative to log $L_E = [\langle a,b\rangle^{20}, \langle a,b,c\rangle^2, \langle a,b,c,c\rangle^1, \langle e,f\rangle^1]$.

and then the resulting models were optimised for extremity by hand. The corresponding metrics are summarised in Table 6.7.

The examples in Figure 6.8 use log $L_E = [\langle a,b\rangle^{20}, \langle a,b,c\rangle^2, \langle a,b,c,c\rangle^1, \langle e,f\rangle^1]$. Note that this log has one frequently occurring trace, $\langle a,b\rangle$, which dominates the probability mass, with some variations, and one completely different trace, $\langle e,f\rangle$. Model 6.8a achieves high adhesion and relevance by covering the main trace and its variations with plausible weights. In model 6.8b, almost perfect adhesion and relevance has been achieved at the cost of simplicity. This model also has perfect control-flow fitness and precision. Trace model 6.8c achieves perfect adhesion and relevance at the cost of poor simplicity.

Flower model 6.8d has poor adhesion and relevance, despite perfect control-flow fitness. The flower model is well-known in process mining as allowing every possible trace for its activities [6, p189 and Fig 6.23]. The many possible alternative traces generated by a flower model mean little probability mass is devoted to either similar traces (for adhesion) or entire traces (for relevance). This flower model has weights taken from the event frequencies in the log (equivalent to the

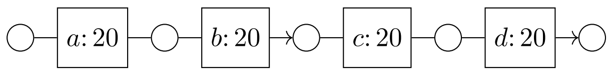


Figure 6.9: Model exemplifying adhesion and entropy variations relative to logs L_F and L_G .
Adhesion+ Relevance- Simplicity+. Partial major trace relative to log $L_F = [\langle a,b,c,d,e\rangle^{20}, \langle a,b\rangle^{1}, \langle b,c\rangle^{1}, \langle c,d\rangle^{1}, \langle d,e\rangle^{1}]$.
Adhesion mid-range(-) Relevance mid-range(+) Simplicity+. Half log coverage relative to log $L_G = [\langle a,b,c,d\rangle^{50}, \langle e,f,g\rangle^{50}]$.

Table 6.7: Quality metrics for paradigm examples in Figures 6.8 and 6.9

	Adhesion			Relevance		Simplicity	
	ADM	TGF5	EM	RDM	HRZ	SDM	$\|F\|$
A+ R+ S++ 6.8a	0.80	0.72	0.91	0.72	2.76	0.67	4
A++ R++ S+ 6.8b	0.83	0.96	0.94	0.74	1.76	0.35	10
Trace 6.8c A++ R++ S-	0.91	1.00	0.91	0.62	1.08	0.18	22
Flower 6.8d A- R mid S mid	0.28	0.00	0.21	0.88	6.37	0.57	29
Partial Major 6.9 L_F Trace A+ R- S+	0.35	0	0.73	0.51	14.42	0.52	8
Half log 6.9 L_G A mid R mid S+	0.65	0.5	0.5	0.73	6.92	0.42	8

Frequency Estimator (see Definition 16), but this has had little quality impact without further structure constraining the space of possible subtraces.

In Figure 6.9, two logs are considered versus the same simple sequence model. In log L_F, a single trace, $\langle a,b,c,d,e\rangle$ accounts for the vast majority of cases (partial major trace). Every event except the last is covered, resulting in high (not perfect) adhesion, but poor relevance. In L_G (half log), there are two frequently occurring traces, one of which matches the model perfectly, and one not at all. As half probability is a state that can minimise entropy, we expect somewhat higher relevance, with at best mid adhesion.

In Table 6.7, we can see how the metrics identified in Sections 6.5.2 and 6.5.3 perform against these extreme cases. To calculate these figures, a modification had to be made to the dimensional realist metrics, as the SSEDC metric returns zeroes for very small logs, such as those explored in this section. We substituted a small log variant which divided model edge count by the product of trace variants and average trace length in the dimensional realist measures, and provide edge count ($|F|$) as a substitute for the native metric.

For Adhesion, the earth movers (EM) and ADM metrics reflected the process edits needed across all models. Trace Generalization by floor (TGF5) generally followed the pattern, but returned zero for the Partial Major Trace scenario, due to the small log.

For Relevance metrics, the Entropic Relevance (HRZ) metric behaves consistently with expectations across the models, though it also shows correlation with Adhesion metrics and makes it difficult to construct a scenario with high relevance and low adhesion. The RDM measure does show high relevance for the Flower model. Though being able to map models to all corners of orthogonal dimensions does meet one goal, the conceptual obscurity may undermine use of this as a productive design constraint for model construction.

For Simplicity metrics, both the edge count and the SDM measure show some consistency with expectations. The SDM measure punishes the perfect model more than the trace model, however, which is due to the influence of factors such as Trace Ratio (TRG2), and again works against intuitive understanding.

6.6 Discussion and Limitations

So far, as shared in Section 6.4, we have seen variations and correlations across the twenty-five metrics collected on 9301 models generated from six real-life logs. These have been analysed quantitatively, identifying the three quality dimensions Adhesion, Relevance and Simplicity. We saw that these components were not associated with known regularities in the experimental inputs, in the form of logs or generation sources. We chose metrics based on the dimensions using two alternative dimensional interpretations. We also investigated example models at the extremes of the dimensions, based on these metrics.

In this section, we discuss the two dimensional interpretations, Dimensional Realism versus Native Metrics, in more detail. We also cover potential applications, and limitations of the current design.

6.6.1 Contrasting Dimensional Interpretations

The two interpretative views we introduce have complementary strengths and weaknesses. Dimensional realism (DR), by treating PCA components as real underlying structures, allows for the revelation of features not directly illustrated by any given metric. Being based directly on the outcome of a principal components analysis, DR is also guaranteed to yield perfectly orthogonal dimensions. Yet that same guarantee, and the “synthetic” nature of PCA, also makes DR dimensions sensitive to the exact metrics chosen as inputs. As new metrics are proposed by the community, or excluded for changing design reasons, the associated dimensions will change.

Native metrics allow for a clearer association between the design concept of a particular metric, the resulting measurement on a specific model, and the dimension it is identified with. However, they lose some information on the underlying quality space, being limited to one metric per dimension. Those metrics also only approximate orthogonality, and may occlude features that DR metrics can indirectly indicate.

6.6.2 Potential Applications

Stochastic phenomena are widespread in the real world, and stochastic models are used widely in settings from Operations Research [160], to healthcare [113] and performance prediction [148].

For stochastic process models specifically, more automated discovery techniques are emerging, but existing metrics for evaluating their quality are not sufficient. To use these discovered models intelligently, more widely applicable metrics, and a better understanding of their meaning and relations, are needed. We envision the dimensions and metrics proposed above can advance this understanding.

It is often necessary in process modelling and mining to choose among potential representations based on a specific use case. For example, very detailed models with lots of elements may be very accurate, but make a descriptive model difficult to explain. The decision is ultimately one of practitioner judgement. Using these dimensions, and their associated metrics, that judgement can now be better informed in a fine-grained way. A practitioner or modeller can decide how much their model adheres strictly to the process it describes, or how much information on complete traces to sacrifice in working with a simpler model. Better tooling can help share this information with users in the right context. For example, an intelligent slider or two dimensional "colour-picker" widget could allow a user to navigate the right level of quality and complexity for their use case. Commercial process mining tools already make use of frequencies. This implicit use of stochastic models could be made explicit, and our work above can help in designing and implementing such features. For instance, there is little point in calculating two highly correlated metrics.

Another strength of stochastic model comparison can be in exploring change in processes over time, a phenomena known as *concept drift*. The Earth-movers' distance metric (EM) has been used to measure this kind of change [30]. Using these approaches, we can imagine a plant manager seeing their manufacturing process has changed, because of the changing importance of existing paths of workflows in the plant, a condition hard to detect without a stochastic perspective. Tools might visualise the impact of the change, for more rapid, productive troubleshooting.

The metrics based on play-out trace probability (Definition 36) may now be calculated on more types of models, and with relatively low computational cost. Our public reference implementation also shows the feasibility of implementing these metrics in industrial tools.

6.6.3 Limitations

Although a wide range of models and logs were used, other datasets may reveal other elements. Larger logs of over 200,000 traces or 16 activities were not used, and SETM use was limited by larger numbers of log activities. The stochastic models used were limited to SLPNs, though some of the discovery models were derived from discovery algorithms with BPMN output, and a mix of discovery algorithms was used. The use of PPTs for random generation and for the seed generation in the SETM limits the possible models generated, though it also constrains them to structured models with consistency constraints on stochastic weights.

The example models in Section 6.5.4 help show the dimensions in use and build an intuition of how these quality dimensions apply in practice. A model with high Relevance and low Adhesion was not identified in this research: the closest was a model with middling to low Adhesion (in Figure 6.9). Such a model would be informative, make the examples symmetrical, and clarify the relationship between different dimensions in practice.

That new metrics changed the dimensional analysis shows the way this empirical work will have to evolve as new data is available. Models discovered from synthetic data can also be used to deepen our understanding of quality measurement of stochastic processes. As an experimentally derived theory, further experimentation will be the ultimate test of generality for all of the proposed dimensions.

6.7 Summary

Organisations may be understood by what they do, and what they do may be described by stochastic process models. To understand the quality of such models, we conducted two experiments studying stochastic process model quality metrics and relationships. Models were generated from six real-life logs and collected using both random model generation and stochastic process discovery. Analyzing a variety of computationally cheap metrics across thousands of models, and metrics from the literature, three quality dimensions were observed with the help of principal component analysis. We named these dimensions Adhesion, Relevance and Simplicity, evolving our understanding of the three dimensions during the course of the experiments and analysis. Based on the analysis, we suggested possible metrics for these dimensions, and showed their use on example models demonstrating their extremes. A number of avenues are open for future work. The methods here suggest extensions of existing techniques, and new implementations of those techniques. Large model datasets may be used to expand the empirical foundations of process mining in other ways. By integrating the dimensions and their associated metrics into process mining tools, further research can be done on use by practitioners. Lastly, it is a spur for the invention of new metrics based on these dimensions, and for the theory to be challenged with further empirical tests.

Chapter 7

State Snapshot Discovery and Qing Civil Service Case Study

The Chinese like to say that history is a mirror. But sometimes, history is a periscope, forcing the viewer to observe at an angle.

Rana Mitter
China's Good War [116]

To date, process mining [6] has been applied to modern organisations such as businesses and governments, particularly those with large quantities of data available in their IT systems. These techniques allow investigators and managers to understand and improve these organisations, often in an industrial or governmental setting. Process mining can also be used to help understand other organisations and questions beyond management science.

In this chapter, we apply stochastic process discovery to a historic setting, to examine a long-lived institution of historical interest, the Qing (1644-1911) Chinese civil service. We use data assembled by historians on civil officials during the 19th century. In such a setting, information systems that log detailed millisecond-precision event data are not available. We present a new type of input log that is more likely to be available in historical contexts, a matching stochastic discovery technique, a visualisation of these models, and an application of our approach to historical data of civil servants' careers in the Qing. We describe this structure as a state snapshot log, and present a new discovery technique, the State Snapshot miner, for constructing stochastic Petri net models from such logs. A case study shows its use in analysing promotion paths for elite graduates in the Qing civil service.

Qing dynasty civil service personnel records allow us to understand a historically important institution in Chinese and world history [68, 47]. China has a long tradition of administration by a formal bureaucracy with well-defined procedures for the appointment, review, and promotion, transfer or termination of officials. From as early as the mid-18th century to the end of the

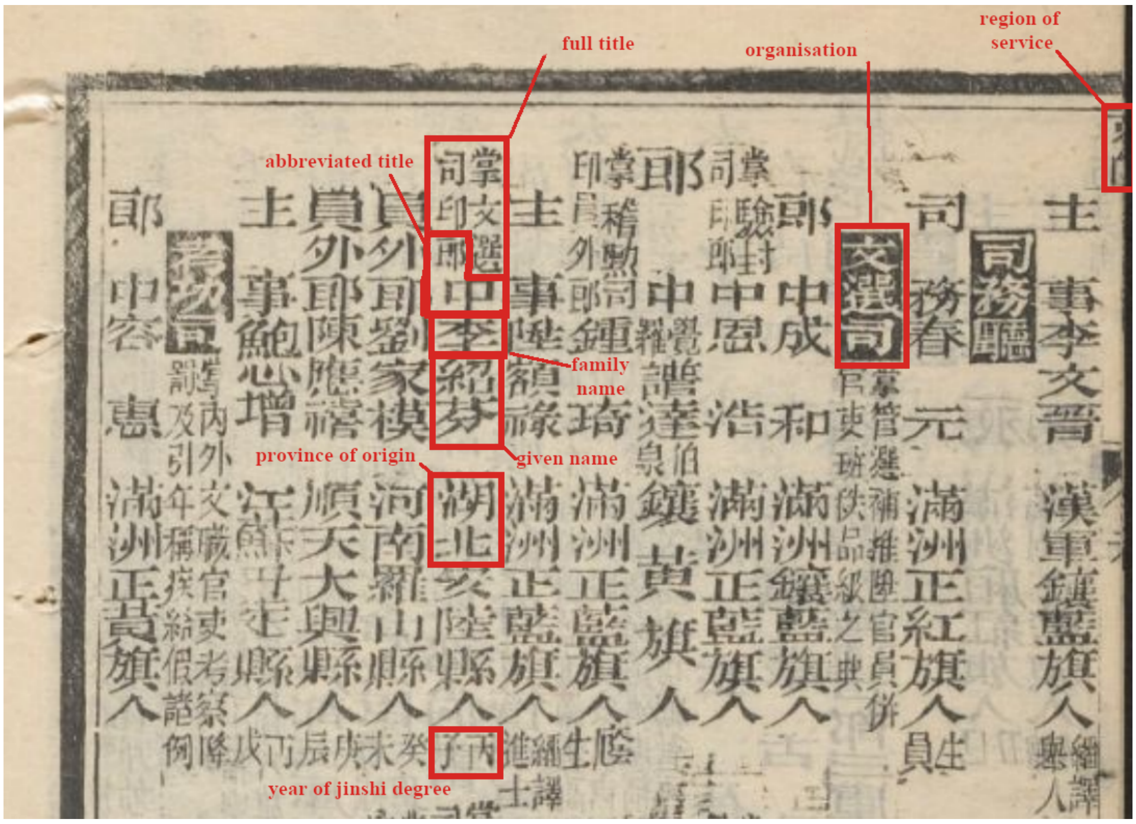

region of service
organisation
full title
abbreviated title
family name
given name
province of origin
year of jinshi degree

Figure 7.1: A civil servant roster *jinshenlu* page from 1897 (volume: Juezhi Quanlan, Guangxu 23, Winter) for a Board Director (郎中), `person_id` 188420067500 in the CGED-Q database. Harvard Yenching Library.

Qing in 1911, the government issued a roster of all regular civil officials, the *jinshenlu* 缙绅錄, every three months. Figure 7.1 shows a sample page from this roster for a Board Director (郎中) named Li Shaofen. It includes the official's name, current roles, qualification, the organisation (e.g. department or bureau) they work in, and other information. An elite official could hold multiple appointments at one time, and these concurrent roles often have mismatched durations. This combination of recording state observations at fixed intervals, and those states holding multiple features from the same feature dimension, is common in other historical sources such as population registers and censuses, due to the expense of recording fine-grained, low latency data.

Social science historians have transcribed available editions of the *jinshenlu* from 1760 to 1911 to produce the China Government Employee Database-Qing (CGED-Q) [40, 47] . Their program of historical research using these and related data to which it has been linked takes "an exploratory and inductive approach to discover basic facts about the Chinese past that are not readily apparent from traditional approaches in history" [47]. They have used the CGED-Q to study time trends in the composition of civil officials and conduct case studies of specific groups of officials.

This chapter focuses on a process mining research question: *How may we discover models of processes from records of concurrent states?* This helps advance historical research examining the career paths of Qing dynasty civil servants, as pursued by professional historians. The application of process mining to historical data not produced by a modern information system, and the investigation of the Qing civil service career paths with a computational analysis of aggregate paths, are both novel.

Our solutions are suggested by the structure of this historical data. Firstly, appointments held by a particular official at a particular date are better thought of as a *state* than an *event*. This marks a change from the currently dominant input for process mining: the event log. We instead characterise our input as a *state snapshot log*, formally defined in Section 7.3.1, where the concepts of cases and traces are retained, but each trace entry is associated with a set of roles. Secondly, the output model must represent concurrent appointments. We use a weighted Petri net where places are labelled, rather than transitions. A weighted Petri net is more concise than a Finite State Automaton (FSA) and represents concurrency more clearly than a Direct Follows Graph (DFG). A new discovery algorithm, the *State Snapshot Miner*, is introduced, which constructs these models from state snapshot logs. These techniques are applied to a 74-year window of civil service appointments, particularly the paths of elite graduates. The analysis confirms and complicates the existing understanding of appointments when compared to a normative model based on the official published appointment rules.

The remainder of this chapter is structured as follows. Section 7.1 reviews existing work specific to this chapter. Section 7.2 lays out formal preliminaries. Section 7.3 introduces state snapshot logs and the State Snapshot Miner for process discovery. The case study on Qing civil servant career paths, and discussion of other applications, is in Section 7.4. Section 7.5 discusses domain expert insights into the solution, and limitations. Section 7.6 summarises the work in this chapter.

7.1 Related Work

This section surveys existing work in computational history and process mining for promotion or career paths, especially with stochastic elements.

In recent decades, historians and social scientists have supplemented traditional methods with techniques that use the power of modern computing to describe, model, and analyse change in societies, a trend which can be loosely grouped under the term computational social science [50]. Social science fields such as economic history have also used new techniques to yield insights from large data sets [78]. One inspiration for this research is computational analyses of organisations in the French Revolution [18], showing identifiable recurring patterns in parliamentary debate during that period. In East Asian history, multi-decade projects to digitise historical sources have made new forms of analysis possible [40, 81]. This chapter uses the CGED-Q [47] database, which collects the Qing dynasty civil service records published in the *jinshenlu*. Translations for Qing civil service names have been taken from an established dictionary and related scholarship on the Qing [85, 46]. Here we focus on a specific subgroup of officials, the holders of the *jinshi* (Metropolitan) examination degree, an elite among officials distinguished by their eligibiity for appointment to high office.

Techniques for unsupervised learning organisational structure from time series data are a form of *organisational mining* [6, p281]. This can include mining organisational models, social network analysis, resource rules or behaviour profiles, as described by a recent survey [159]. In this sub-field, data attributes associated with operational activities are used to compute models of the organisation that performs these activities. In the current study, organisational models are constructed, but the available data is not on operational civil service activities, such as the judgements of local magistrates, or the reports of central government investigators, but on posting and promotion data itself. This data allows for the construction of a different type of organisational model, which describes career paths in terms of stochastic control-flows.

Life courses, including career paths, are a well established topic in social science research [54]. Sequence analysis methods [10, 54] identify patterns of life stages in a cohort. Sequences are compared using similarity metrics in order to identify clusters or typical paths. In process mining terms, quantitative similarity metrics are calculated by pairwise trace comparison, including using string edit distance, as in the Earth movers' distance measure (EM) [95]. Unlike process mining, sequence analysis does not construct a workflow model, or visualise one. Sequence analysis has been used on contemporary and historical data sets. A study of German musicians 1660-1810 [10] identified a number of typical church musician career paths. Like the Qing officials in this chapter, musicians could hold simultaneous roles. There, simultaneous roles were excluded; here, our discovery technique accommodates such roles. The sometimes long and complicated career paths of Qing officials have also motivated the development of an interactive browser for CGED-Q data [156], including visualisations of cohort and group mobility; workflow models, as in process mining, would be a complementary addition.

Though the lessons of Roman history for process science have been discussed in scholarly speeches [128], to our knowledge, process mining techniques have not previously been used on datasets from before the advent of modern information systems. Process mining has been applied to modern career paths. An analysis of 35 years of data on employees at a Portugese public organisation [137] found process mining less than insightful, while noting that employees at that organisation rarely changed roles. Direct Follow Graphs (DFGs) were used, and job roles were equated with DFG transitions, on a 585 person data set. University student career paths were used for experiments in declarative process mining [21], using a five-year, 813 student data set from an Italian university. A discovery technique constructed predictive models consisting of both logical rules and associated probabilities, given a conventional event log input.

The State Snapshot Miner that we introduce in this chapter creates models in the form of a stochastic Petri net variant. Other forms of stochastic process discovery are designed to work with event logs, such as estimation on a control-flow model as in Chapter 3, reduction on weighted process trees as in Chapter 4), Stochastic Labelled Petri Nets with dynamic weights [99], or estimation with input place-transition Petri nets [130] or Data Petri Nets [112]. The State Snapshot Miner, by contrast, is created to deal with the challenges of state snapshot logs and concurrent roles.

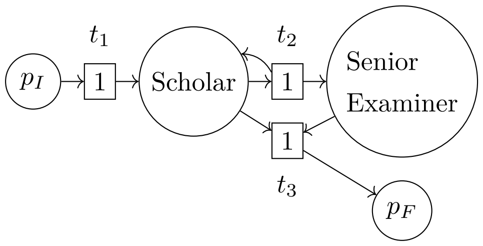


Figure 7.2: Example Place-labelled Petri net.

7.2 Preliminaries

A role is an observable element performed in a process. The set of all roles is $\mathcal{R}$. In this chapter roles are restricted to natural language strings describing a civil service appointment, such as "Senior Examiner".

Definition 40 (Place-labelled Petri net)**.** *A* place-labelled Petri net *is a directed graph* (P, T, F, W, M_0)*, where* (P, T, F, M_0) *is a place-transition net as in Definition 4. Each place has a capacity of one token.* $W : T \to \mathbb{R}^+$ *gives weights for each transition. As for place-transition nets,* marking $M \in \mathbb{P}(P)$ *is a set of places with tokens. A transition is* enabled *when every incoming place has a token and, if those incoming tokens are removed, every outgoing place does not have a token. Let* T_e *be this set of enabled transitions for a marking* M*.*

$$T_e = \{t \in T \mid \bullet t \subseteq M \wedge (t \bullet \setminus \bullet t) \cap M = \varnothing\}$$

The probability of an enabled transition $t \in E$ *firing is then given by* $\frac{W(t)}{\sum_{t' \in T_e} W(t')}$*.*

The set of all Place-labelled Petri nets is denoted $\mathcal{PLN}$. In this chapter, places Place-labelled Petri nets are simply roles; ie $P \subseteq \mathcal{R}$. An example Place-labelled Petri net is in Figure 7.2. When a single token is in the Scholar place, the t_2 transition is enabled. When fired, the transition produces a token for Senior Examiner and returns a token for Scholar. The new marking of { Scholar, Senior Examiner } does not allow the t_2 transition to be enabled again, as both the Scholar and Senior Examiner output places already hold tokens.

7.3 Discovery With the State Snapshot Miner

In this section we introduce a process discovery technique, the State Snapshot Miner, and discuss its implementation, applications and limitations. The miner takes state snapshot logs as an input, which are also introduced formally.

7.3.1 State Snapshots

The set of all case identifiers is represented as $\mathcal{U}_{case}$, and the set of all times as $\mathcal{U}_{time}$. A state comprises a case identifier, a timestamp, and roles.

Definition 41 (State)**.** *A* state s *is a tuple* $(c, t, z) \in \mathcal{U}_{case} \times \mathcal{U}_{time} \times \mathbb{P}(\mathcal{R})$*.*

A state models the way officials may hold one or more bureaucratic positions at any moment in time in this study. In other domains, it could model a network of Internet-Of-Things (IOT) temperature alarms, which can alert both that temperature exceeds a threshold and that their battery is running low. Or it could model equities market data, which for some exchanges can simultaneously report a price, an auction state, and a halt state for a given stock at a given time.

A time series of states forms a state trace.

Definition 42 (State trace)**.** *Let $R \subseteq \mathcal{R}$ be a set of roles. A* state trace $\sigma \in \mathbb{P}(R)^*$ *is a sequence of role sets.*

A log collects traces. The time attribute of the state is used to order the sequence.

Definition 43 (State snapshot log)**.** *A state snapshot log L is a multiset of traces: $L \in \mathcal{B}(\mathbb{P}(R)^*)$.*

The set of all state snapshot logs is $\mathcal{SL}$. A state snapshot log is constructed from a set of states by creating a trace for each unique case id, ordered by timestamp, and collecting them in a multiset.

Table 7.1: Example state log excerpt, SL_E, from CGED-Q data.

person id	name	date (y-m)	roles
188330054900	陳冕 Chen Mian	1883-06	{ 修撰 Senior Compiler }
189230033500	劉福姚 Liu Futiao	1892-09	{ 修撰 Senior Compiler }
189230033500	劉福姚 Liu Futiao	1893-03	{ 修撰 Senior Compiler, 正主考 Chief Examiner }
189230033500	劉福姚 Liu Futiao	1894-03	{ 修撰 Senior Compiler }
189230033500	劉福姚 Liu Futiao	1897-03	{ 修撰 Senior Compiler, 副考官 Vice Examiner }

As an example, consider the log excerpt in Table 7.1. This holds a sample extract for two officials who were first place, in different years, in the elite palace examination. Note that multiple roles may be held by one official at a particular date. For brevity, only the first entry for new role combinations are included.

7.3.2 Discovery

In general, state snapshot discovery is a function $\mathcal{SL} \rightarrow \mathcal{PLN}$ transforming a log into a model. We introduce one such algorithm, the State Snapshot Miner. Some straightforward functions for calculating direct follows relations are defined, followed by the algorithm itself.

The *dff* function calculates direct-follow frequencies for a state snapshot log with roles R. It uses subsequence frequency function sqct from Section 2.1.

$$
\begin{aligned}
dff &: \mathbb{P}(R)\times\mathbb{P}(R) \times \mathcal{SL} \rightarrow \mathbb{N} \\
dff(x,y,L) &= \sum_{\sigma \in L} \mathrm{sqct}(\langle x, y\rangle, \sigma) \cdot L[\sigma]
\end{aligned}
$$

The *dh* function gives frequencies for initial roles and the *df* function for final roles.

$$\begin{aligned} dh, df &: \mathbb{P}(R) \times \mathcal{SL} \to \mathbb{N} \\ dh(x, L) &= \sum_{\sigma_1 \in \{\langle x \rangle + \sigma_2 \in L\}} L[\sigma_1] \\ df(x, L) &= \sum_{\sigma_1 \in \{\sigma_2 + \langle x \rangle \in L\}} L[\sigma_1] \end{aligned}$$

The miner can then be defined.

Definition 44 (State Snapshot Miner)**.** *Given a log $L \in \mathcal{SL}$ over roles $R \subseteq \mathcal{R}$, the function ssm outputs a place-labelled Petri net model.*

$$\begin{aligned} ssm(L) = (P, T, F, W, M_0) \ &\textit{where} \\ p_I \notin R \wedge M_0 &= \{p_I\} \ \textit{an initial place} \\ p_F \notin R \wedge p_F &\neq p_I \ \textit{a final place} \\ P &= \{p_I, p_F\} \cup \{a \in R \mid \exists \sigma \in L, s \in \operatorname{ran} \sigma \bullet a \in s\} \\ T_I &= \{s \in R \mid dh(s, L) > 0\} \ \textit{initial transitions} \\ T_F &= \{s \in R \mid df(s, L) > 0\} \ \textit{final transitions} \\ T_R &= \{(s_1, s_2) \in R \times R \mid dff(s_1, s_2, L) > 0\} \\ T &= T_I \cup T_F \cup T_R \\ F_I &= \{p_I \mapsto t \mid t \in T_I\} \cup \{t \mapsto p \mid t \in T_I \wedge p \in t\} \\ F_F &= \{t \mapsto p_F \mid t \in T_F\} \cup \{p \mapsto t \mid t \in T_F \wedge p \in t\} \\ F_R &= \{p \mapsto (s_1, s_2) \mid (s_1, s_2) \in T_R \wedge p \in s_1\} \\ &\quad \cup \{(s_1, s_2) \mapsto p \mid (s_1, s_2) \in T_R \wedge p \in s_2\} \\ F &= F_I \cup F_F \cup F_R \ \textit{arcs follow role set pairs} \\ W &= \begin{cases} dh(t) \ \textit{if} \ t \in T_I \\ df(t) \ \textit{if} \ t \in T_F \\ dff(t) \ \textit{if} \ t \in T_R \end{cases} \end{aligned}$$

The miner works as follows. Each role becomes a place. Initial and final places are added to these. Transitions correspond to either sets of roles (for initial and final transitions), or pairs of sets of roles. Flows then join transitions going from one set of roles to another, as indicated by the evidence in the state snapshot log. There is a separate transition for each distinct combination of input and output role sets, and ones to join the initial and final places, for starting and final role sets. The weight of transitions reflects how often this distinct combination occurs. Time intervals where no role is reported for a given case are treated as continuations of the previous model state (marking), unless they are after the end of the sequence (terminal).

Applying the State Snapshot Miner to SL_E in Table 7.1 yields the model in Figure 7.3.

The time complexity of the State Snapshot Miner is linear in the number of state snapshots and the number of roles. Each state snapshot must be considered in evaluating initial, adjacent, and final activity sets. There are $2^{|R|}$ possible activity sets in a log, and so $(2^{|R|})^2$ possible activity set combinations in the worst case. At each iteration through a trace entry, some data structure indexing activity set combinations must be updated with frequencies. This data structure (say, a hashtable or tree) can be indexed in log time. The complexity is then

$$\begin{aligned} O(||L|| \cdot \log((2^{|R|})^2)) &= O(||L|| \cdot \log 2^{2|R|}) \\ &= O(||L|| \cdot 2|R| \log 2) \\ &= O(||L|| \cdot |R|) && \text{as } 2 \cdot \log 2 \text{ is a constant} \end{aligned}$$

To allow managing the complexity of output model visualisations by abstracting away from infrequent paths, noise reduction is added to this algorithm by pruning transitions below a weight threshold, and the arcs that connect them.

7.3.3 Implementation and Visualisation

The State Snapshot miner has been implemented in Python[1]. The output Place-labelled Petri net models are exportable to PNML or as pm4py objects. It includes optional noise reduction by transition weight, taking a threshold interpreted as a proportion of the sum of all direct-follow role set pairs (i.e., transition weights).

The visualisation for Place-labelled Petri nets developed for this project uses conventional ovals for places and rectangles for transitions. Arc width is scaled with transition weight, and place size scaled with role frequency. Visualisation support allows the inclusion or suppression of final places.

[1]Source and sample data are at `https://github.com/adamburkegh/statesnap-miner`. See Appendix A.

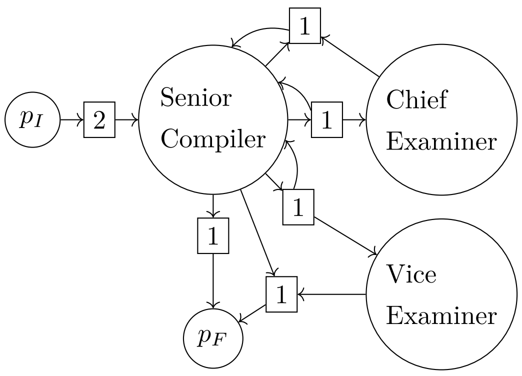


Figure 7.3: Model for two sample officials career paths (SL_E) discovered by the State Snapshot Miner.

7.4 Qing Promotion Paths Case Study

These techniques have been applied to data from the CGED-Q database to examine civil service career paths.

7.4.1 Background

The Qing state had rules for appointments to civil service positions. Notably for our study, many officials were appointed based on performance in civil service exams. There were multiple stages of examination, starting from a prefectural exam, through a provincial exam to national (or Metropolitan) and palace exams. The exams were highly selective, with approximately 2% of the adult male population, or 2 million people, presenting for each prefectural examination, but only 26000 candidates succeeding at the national exam over the entire course of the entire Qing dynasty [68]. Success at each level of these exams led to the award of a degree, with admission to the palace exam granting the highest degree: *jinshi*. Specific rules applied to the appointment of the *jinshi* degree holders, which have been the focus of previous scholarship [68, 47]. These defined a “fast track” of sorts for these elite candidates, where they could be quickly placed in relatively senior roles.

The CGED-Q, sourced from the original *jinshenlu* records, holds a rich collection of demographic and organisational detail. Figure 7.4 is a model in Object-Role Modeling (ORM) notation [80] mapping key entities and relationships. Here we can see familiar personnel management concepts such as family name or position, those with analogues in modern universities such as very precise degree descriptions, and concepts more specific to the Imperial Chinese context, such as the courtesy name (自號).

The flow-chart in Figure 7.5, adapted from [46, p96], summarises the normative process model of this appointment and promotion process. Though the candidates had already succeeded at three exams, a further ranking was provided by a palace exam, hosted by the emperor himself. Candidates were ranked into three tiers, with the top three candidates constituting Tier 1 (一甲). These three candidates, and sometimes the top placed candidate in Tier 2, were given posts at the Hanlin Academy and groomed for high office. The Hanlin Academy was an elite academy in Beijing that served the imperial court. The remaining Tier 2 and 3 candidates sat a further exam, the court exam, which determined admission to the Hanlin Academy, or appointment to less prestigious, but still important, roles in the capital and the provinces. In Figure 7.5 we can see, for example, that the top candidate in the palace exam was placed into a Senior Compiler post (修撰), part of a team which drafted and compiled official histories, including the dynastic history. Being the top graduate earned a specific title (状元) and the distinction was noted in the *jinshenlu* and other records.

7.4.2 Data Preparation

To examine elite graduate Qing civil service careers from a process perspective, we used an extract from the CGED-Q database that covered ethnic Han officials who earned the *jinshi* degree and sat the palace exam over the period 1830-1904. The CGED-Q records are most complete after 1830,

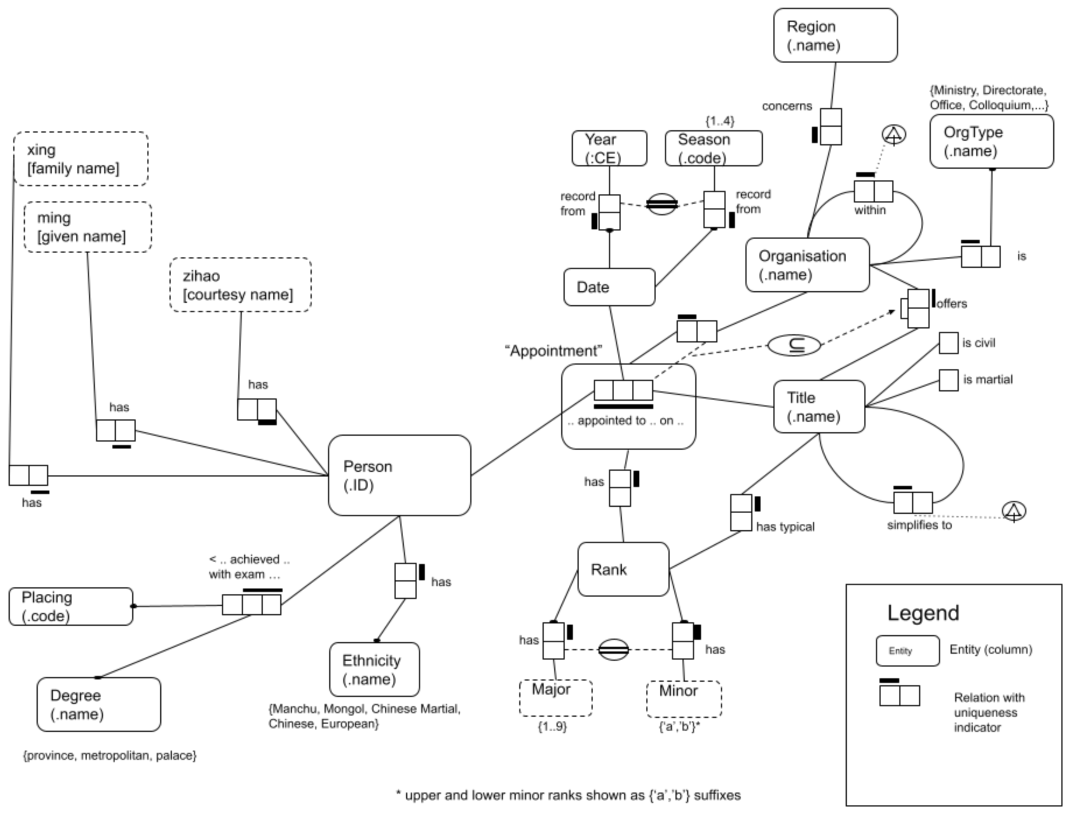


Figure 7.4: ORM conceptual schema for Qing Civil Service Promotions data held by the CGED-Q.

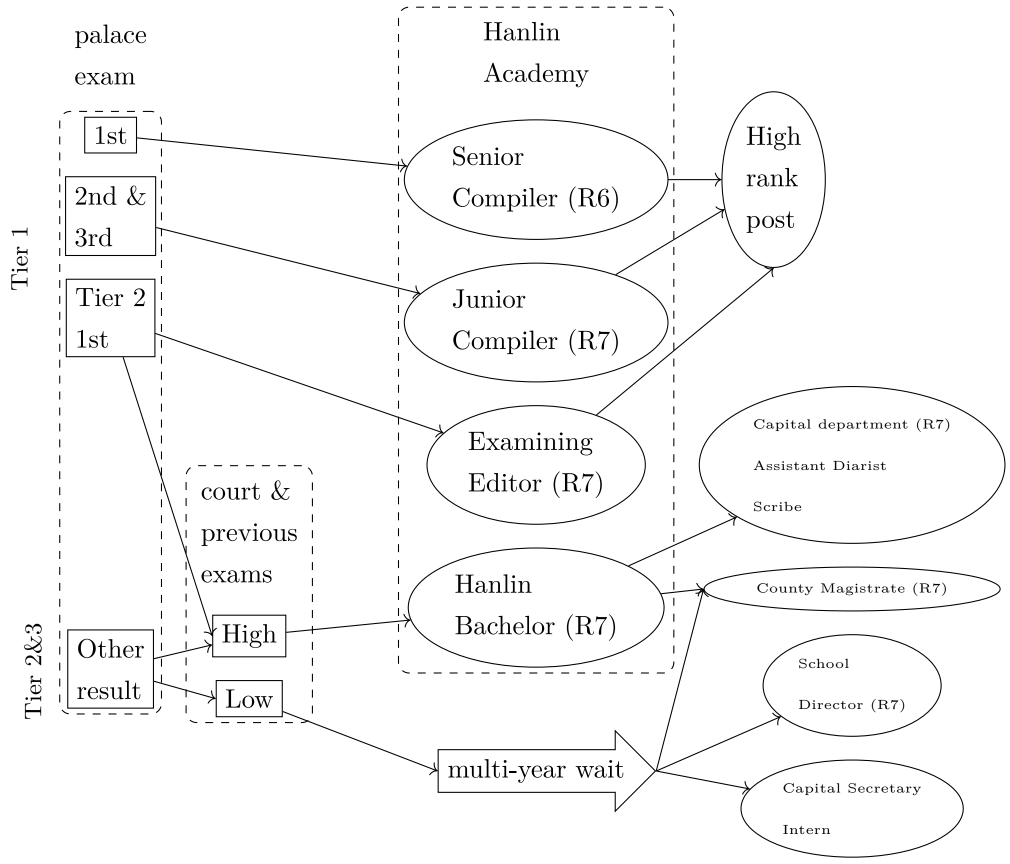


Figure 7.5: Flow chart of appointments for *jinshi* 進士 degree holders after taking the palace exam, adapted from a recent study [46, p96]. Ranks are shown by (Rn), with R1 being the most senior.

and the exams were abolished in 1905. This period was a turbulent and eventful one in China, encompassing the Taiping Rebellion and both Opium Wars. In the current version of CGED-Q, the challenge of uniquely identifying officials over time, using a combination of name and other identifiers, has been most completely solved for ethnic Han officials, in part due to the relative uniqueness of Chinese character names in the period [39]. CGED-Q has established a unique `person_id` field as a result of this work. We joined the *jinshenlu* extract with the examination records from the same period (會試題名錄) to obtain the palace exam tier. Officials who were already mid-career in 1830 were excluded.

Table 7.2: CGED-Q Log Subset Statistics, 1830-1904

Palace Exam Filter	Year Filter	Events	Cases	Max Concurrent Roles
Top place	None	375	36	18
Top place	3	64	36	3
Tier 1 & 2 Top 7	None	11790	3897	5
Tier 1 & 2 Top 7	3	5299	3897	4

Concurrent roles for an official were sometimes represented as separate records in the same circular, and sometimes concatenated in a single field in the *jinshenlu* record. This is an instance of the *Distorted Label* event log imperfection pattern [143], a variant extreme enough to result in role conflation. To resolve this, the role was split according to a new project dictionary of known roles. Concurrent roles often had partially overlapping time periods and different durations. For example, an official may have a permanent posting, which lasted a number of years, a temporary appointment, lasting a few months, and an honour or title which endured.

Honours, such as “gifted the peacock feather” (賞戴花翎), or “member of the Hall of Literary Profundity” (文淵閣校理) were similar to knighthoods in the British system. A number of synonyms for roles were also remapped to a common role name, in an instance of *Synonymous Labels* [143]. For example. those allowed to study at the Hanlin Academy (Hanlin bachelors 庶吉士) often had the province name or graduating class name included in their role title, and this was removed. Chronological gaps in sequential appointments, which might indicate unemployment, were not distinguished from other sequences of states in this log. For elite early career officials in this period, unemployment would be rare, but gaps of approximately six months do exist in the CGED-Q database, when the original circular for that season was not available for entry into the database. Lastly, continuing role combinations were conflated into a single record, and this was formatted as a comma separated value (CSV) file that could be parsed as a State Snapshot Log as described in Definition 43. The `person_id` was used as a case id, the job title as the role, and the year and month as a timestamp. The resulting log consists of 219276 individual appointment records. Table 7.2 lists statistics for key subsets of this log.

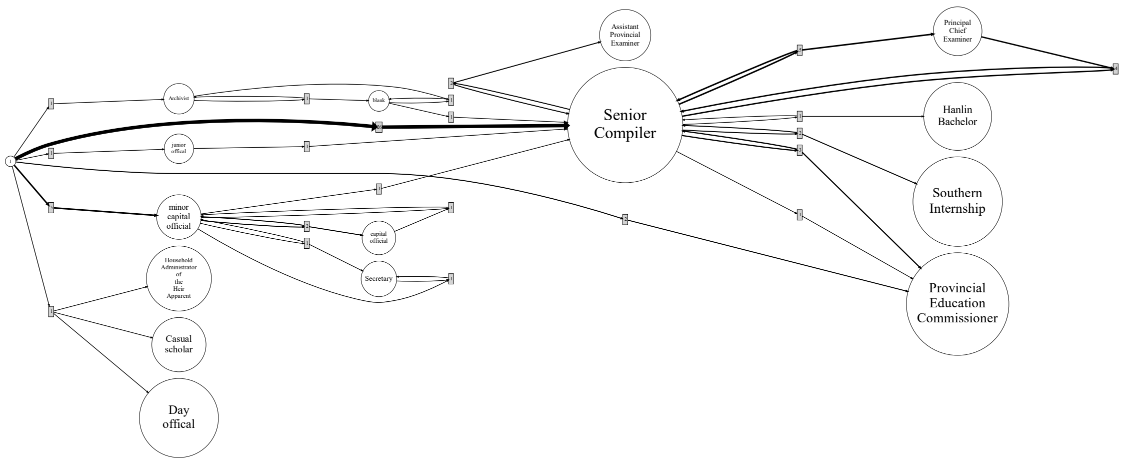


Figure 7.6: Roles held by officials who placed first in the palace exam (状元), in the first three years of their career, 1830-1904.

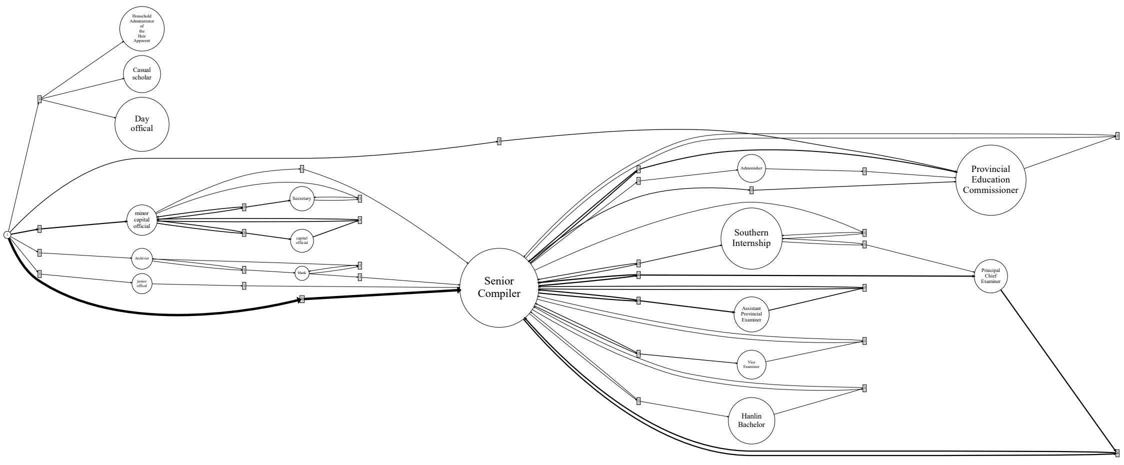


Figure 7.7: Roles held by officials who placed first in the palace exam (状元), in the first five years of their career, 1830-1904.

7.4.3 Early Career Path Models

A variety of models based on different data selections and periods of appointment were produced for analysis and discussion during the study. Some example insights into the domain gained with the State Snapshot miner can be explained using the models in Figures 7.6, 7.7, and 7.8. The state log was filtered by time to produce logs for the first few years of an official's career. The State Snapshot Miner was then applied to a variety of filtered subsets for different exam tiers and career segments. All diagrams for this case study exclude the final place, and the transitions leading to it (T_F), for clearer visualisation. Arcs and places are also scaled by frequency. Conceptually, given the domain, all career stages are potentially final markings. The undisplayed transitions and weights are available in the backing Place-labelled Petri net model.

Figure 7.6 is a model of the promotion paths for the top-placed candidate in the palace examination over the first three years of their career. Thirty-six officials are in this category, allowing for fine-grained models. We can see that this confirms the normative model in Figure 7.5, showing these officials appointed as Senior Compilers within three years, and often earlier. One exceptional official was instead appointed to a privileged position as Household Administrator of the Heir Apparent, still fulfilling the promise of a fast track. There are also elements not shown in the simplified normative model, such as officials serving concurrent roles as provincial interns or examiners. Such officials sometimes served briefly in junior roles before starting their Senior Compiler appointment. In Figure 7.7 shows the first five years after graduation for these candidates, with more taking on temporary examination roles in parallel with their Senior Compiler appointment.

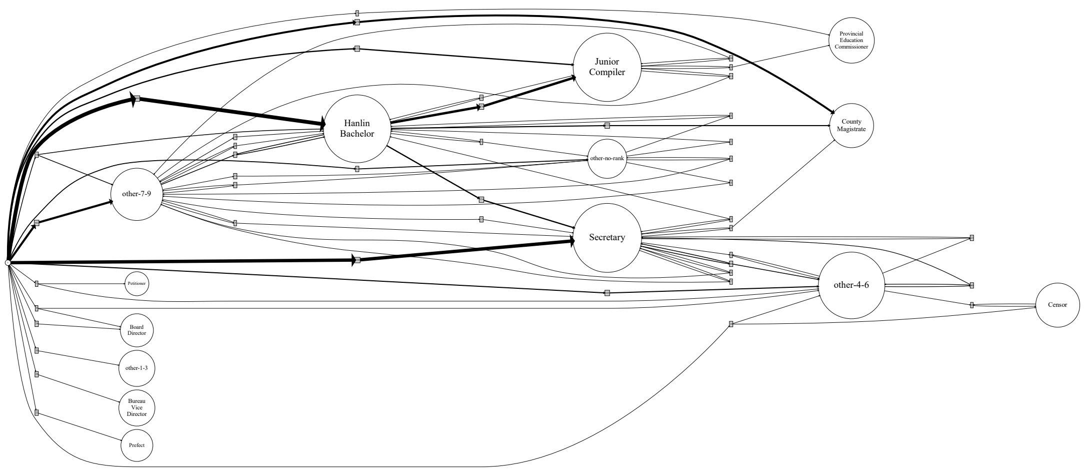


Figure 7.8: Roles held by officials placed in Tier 1 or 2 in the palace exam, in the first three years of their career, 1830-1904. Top seven roles with rank conflation on remainder, noise reduction 0.08%.

Figure 7.8 is a model of the 3897 officials achieving Tier 1 or 2 in the palace exam, over the first three years of their career. To obtain a comprehensible model, the seven most popular roles were retained, and the remainder replaced with their civil service rank. Ranks in CGED-Q are identified at a three band granularity, e.g. 4-6, so these conflated roles are shown as other-4-6. Noise reduction of 0.08% was also applied. Figure 7.5's normative appointment model reflects known regulations for appointment, but the degree to which these rules were actually adhered to is not well established. The Figure 7.8 model largely confirms the normative paths were respected, while showing a number of idiosyncratic variations occurred in practice. A number of Tier 2 officials are admitted to the Hanlin Academy as Hanlin Bachelors. 365 proceed onto Junior Compiler positions, while many others are appointed as Secretaries. Those appointed as County Magistrates or Prefects do not move beyond that within this two-year period. Lastly, more senior (and diverse) rank 4-6 openings were possible directly out of the palace examination without necessarily placing in the first tier.

For these logs, execution of the miner completes in seconds on a Windows 10 machine with a 2.3GHz CPU machine, 10 Gb memory, and Python 3.8.

7.5 Discussion

In this section we discuss insights shared by our historian collaborators, and the design and limitations of the miner and formalism.

7.5.1 Domain Expert Insights

Development of the data preparation, discovery algorithm, and visualisation was an iterative and collaborative process in a team including both technologists and Qing historians, some of whom were already familiar with computational data science tools, and some of whom who were relative novices. We conducted structured reflections in the team to consolidate our understanding of the impact and potential of these process mining techniques.

In general, the historians understood the meanings of directed graph visualisations quickly. Historians also contributed design suggestions on visual elements that were incorporated in the tool, and felt part of the design process. The (conventional) Petri net visual representation of concurrency, though understandable, was also noted as complex or “cluttered” when temporary and long-term roles were in parallel.

The value of comparison with the normative model in Figure 7.5 was highlighted by historians, and seen to complement existing domain methods. “There’s a lot of things in eighteenth and nineteenth century China where there’s deviation between the regulations and the actual practice, so actually showing that practice largely followed the [...] regulations [...] is certainly quite important.” The identification and visualisation of typical versus exceptional pathways was seen as a particular strength.

It was noted that Tier 1 officials, such as the subset in Figure 7.6, have already attracted significant scholarship using qualitative close reading techniques. Models from larger data sets, such as the Tier 1&2 combined model in Figure 7.8, were reported as new. “When you get into categories of people that are too numerous to subject to case studies, then it really gets interesting to have that [...] zoomed out summary view.” They also see the techniques as having potential for discovering hidden rules for appointments that are not documented in official manuals.

7.5.2 Limitations

The current miner does not support multiple places having the same role, or silent places with no observed activity.

The miner gives a way to represent models in a diagram with a well-defined formal semantics derived from stochastic Petri nets. Other representations are possible. For example, a decision tree, or Stochastic Deterministic Finite Automata (SDFA) [151], could also represent all of the possible states for a set of official careers. In the case of a decision tree, representing the variation in paths through particular roles would result in a large number of entities. For an SDFA, the number of entities would be smaller, but each unique state would have to correspond to multiple official appointments. Though the state space is finite, its size would still explode as roles and path variations increased. A similar problem applies to Direct Follow Graphs. A Petri net, where the state is represented by a marking, allows for a more concise representation of both state and concurrent roles. Our historian collaborators were able to understand the diagrams with only a basic explanation, possibly because places are directly identified with roles.

A limitation on the accuracy of the algorithm comes from using direct-follows frequencies to provide weights. This can over-represent the probability of such transitions in loops. E.g. the trace $\langle \{b\}, \{b\}, \{b\} \rangle$ will contribute 2 to the weight of the $(\{b\}, \{b\})$ transition, and the b place will occur in a loop, causing a "double counting" effect.

The domain insights of this tool are limited by the degree of data preparation on these historical sources. For example, the inner join of career records with examination records excludes officials not in both sources, and there were records in this category. Tier 3 palace examination candidates, and the top Tier 2 candidate, were descoped from this promotion study due to time constraints. It is a general rule that the quality of data in a source is a function of how it is used. As this is the first use of this data for process mining, significant effort was required to reorganise the data into a suitable activity log shape. Future work that looks at other historical domains, or even other types of CGED-Q records, will also require time investment in data preparation.

7.6 Summary

The Qing civil service case study shows that process mining concepts and techniques can be applied well beyond modern information technology settings. Process mining traces its antecedents to an industrial process improvement tradition, including the scientific management of Frederick Taylor, the assembly line of Henry Ford, and the twentieth century advent of modern computing [6, p55]. The Qing civil service, and its records, are from a different culture that predates all of these social changes. Historical, paper-based bureaucracies are also information technologies, and the behaviour of an organisation may still be understood through its processes, especially if the records are available in a structured digital form.

The setting still presented significant research challenges arising from the structure of the data. To meet these, we have introduced state snapshot logs, a discovery algorithm, the State Snapshot Miner, for discovering stochastic process models from such logs, an implementation, and a visualisation of the resulting models. Models of the early career paths of elite officials show conformance to the bureaucratic rules of appointment, and variation and complications in how such rules were followed in practice. For instance, promised appointments may be awarded after a short period in less prestigious roles, even after an official had achieved exceptional performance in examinations. These insights are difficult to obtain using qualitative close reading methods, which focus on individual officials, or quantitative statistical methods insensitive to paths.

Looking forward, state snapshot mining can also be used in a more exploratory mode, especially if the tools are extended to allow interactive data selection, constraint, and filtering. Other uses of the miner might include Internet of Things (IOT) sensor data, where multiple sensors capture the state of a system at intervals. The use of process mining techniques on historical data can be extended using the formalisms introduced in this chapter. More broadly, progress by historians applying computational and quantitative methods has made many historical datasets available, and we see understanding this "long history of information systems" as a rich and promising area of research.

From the perspective of stochastic process mining, this chapter explores structures which are not simple “overlays” of transition weights on control-flow models produced by existing discovery techniques. In the terminology introduced in Chapter 3, the State Snapshot Miner is a *direct discovery* algorithm, as it is not decomposed into separate control-flow and estimation steps. The stochastic model elements were of immediate interest to our domain expert partners, who wanted to identify normative paths and “modal pathways” in their data. This chapter also explores how visualisations may complement the stochastic elements of models, based on expert user feedback.

Place-labelled Petri nets may have further extension using duplicate-labelled places, or combining labelled places and labelled transitions, for example where some states and some events can be identified from sequential data. The use of such models would also benefit from a calculation for trace probability.

Chapter 8

Conclusions

I am a theater of processes, he told himself.

Frank Herbert
Dune [82]

In the course of this thesis we have investigated *how processes in organisations can be understood using stochastic models mined from sequential data.* We have introduced a number of techniques to *discover* labelled stochastic net models (RQ1), and to answer *conformance* questions on such models (RQ2), specifically, calculating *trace probability* (RQ2.1) and articulating process model *quality dimensions* (RQ2.2). We also stated solution criteria. This chapter concludes the thesis, summarising solutions in Section 8.1, and reviewing against solution criteria in Section 8.2. Section 8.3 considers limitations, and Section 8.4 discusses potential future work.

8.1 Summary

Chapter 3 addressed the question of discovering stochastic process models when control-flow models already exist, RQ1.1. A new framework for *weight estimation* was introduced, as well as six new *estimators* for producing Stochastic Labelled Petri Nets (SLPNs) from log and control-flow model inputs. The estimators were competitive with existing public implementations in experimental trials, and returned meaningful results on a broader range of input logs and models.

Chapter 4 addressed direct discovery of stochastic process models from logs, without an intermediate control-flow model. To do so, it described a new formalism, *Probabilistic Process Trees (PPTs).* These are weighted process trees with probability semantics and weight constraints, and a translation to SLPNs. Formal properties of PPTs were explored, and used in introducing a new framework for discovery algorithms, the *Toothpaste Miner* framework. These apply reduction rules with known quality properties until a final model is output when no more rules can be applied, and execute in polynomial time. In experimental trials against other discovery algorithms, including those in Chapter 3, Toothpaste Miner achieves good consistency and quality, trading this off against model simplicity.

Chapter 5 used PPTs for a novel solution to the conformance problem of *trace probability* (RQ2.1), Trace-Prob. Exact closed-form solutions on some classes of PPT were shown, as well a solution on all PPTs to within a parameterised approximation bound.

Chapter 6 addressed the conformance problem of *quality dimensions* for stochastic process models (RQ2.2). An experimental dataset was constructed based on thousands of models built from six public logs, existing and exploratory metrics, and a variety of discovery algorithms. These included discovery techniques in Chapters 3 and 4. Analysis, including of principal components, then suggested three quality dimensions: *Adhesion, Relevance, and Simplicity.* These dimensions were explored with example models, and metrics that to approximate these dimensions were proposed. This research also resulted in secondary contributions of interest. Firstly, a discovery algorithm suitable for laboratory work, the *Stochastic Evolutionary Tree Miner (SETM)*, makes use of the PPTs from Chapter 4, and provides another solution for RQ1. Secondly, a play-out technique for log generation from SLPNs, which is also an *approximation function for stochastic languages*, relating to trace probability and RQ2.1. Lastly, *exploratory metrics* were introduced that may be the basis for future conformance work (RQ2).

Chapter 7 challenged the theory and practice of stochastic process mining with a new data source from the Qing dynasty civil service. The use of process mining on historical data, predating modern IT systems, was novel. In working with this data to map career paths for elite officials, we found an expanded idea of sequential data logs was needed. The record of multiple parallel appointments was conceptualised and formalised as a *State Snapshot Log.* To discover career path models using such logs of concurrent states (RQ1.3), a new discovery algorithm, the *State Snapshot Miner*, was introduced. This produces a Petri net, with stochastics defined by transition weights, and where roles are identified with places rather than transitions. The resulting models were used by historians of the Qing dynasty to validate and contrast with the official appointment rules for appointing new elite graduates.

8.2 Evaluation Against Solution Criteria

In the introduction, in Section 1.3, we suggested success criteria for this research: that it should be suitable, conceptual, empirical, formalised, input general, and tractable. In this section we demonstrate how the criteria have been met. We also comment on discovery algorithm maturity specifically, using suggested maturity levels DM1-DM4 [158], also discussed in the introduction.

C1 Suitable. Process mining for control-flow models has made extensive use of labelled Petri nets [6] as a formalism, as it combines deeply investigated mathematical properties with a straightforward diagram. This research has followed other stochastic process mining literature [130, 95, 101] in using labelled stochastic nets, mostly in the form of SLPNs (Chapters 3, 4, 6). This allows solutions from this work to be composed and extended with other techniques in the field. The new formalism of PPTs, used in Chapters 4, 5 and 6, are also translatable to SLPNs, and connect to the existing process mining formalism of process trees. In Chapter 7, a different form of labelled stochastic net is employed, which is still in the family of Petri nets. These structures then facilitated the computation of discovered process models, trace probabilities, and the investigation of quality dimensions. For discovery algorithm maturity, suitability (**C1**) is also a component of design quality (DM1), and also helps to illustrate effectiveness (DM4). The detail of the DM4 criteria is stricter than application in a domain, however. It requires demonstration of monotonic quality improvement on quantitative benchmarks when challenged with real-life logs for which the underlying process is unknown. This level of maturity has been difficult to demonstrate. In stochastic process mining, discovery techniques, metrics, metric relationships, and the quality concepts benchmarks represent are all maturing, a situation this research itself has tried to advance. Without such a demonstration, these discovery techniques do not meet DM4.

C2 Conceptual. New solutions are introduced in each of the research contributions of this thesis. Mathematical formalisms (**C4**) have been used to abstract algorithms and techniques away from domain or technology-specific aspects, such as hospital routines, or particular programming languages. Where new concepts have been introduced, they have been generalised where appropriate. Weight estimation (Chapter 3) and Toothpaste Miner (Chapter 4) discovery techniques are frameworks with multiple instantiations, and potential future extensions that fit within those frameworks. The introduction of PPTs was sufficiently general to be used across multiple domain event logs and process mining problems, and Java and Haskell implementations, in Chapters 4, 5 and 6. The Place-Labelled Petri Nets and State Snapshot Miner introduced in Chapter 7 also abstract away from the detail of Qing dynasty society to more general problems of reasoning over sequential parallel states. The quality dimensions in Chapter 6 are the result of a bottom-up analysis from experiments on real-life data, but they are also a summarisation across multiple problem domains using terms intended to invite broader process mining use.

C3 Empirically Grounded. Real-life logs are used throughout. Public real-life event logs are used in investigating weight estimation discovery, the Toothpaste Miner, and analysis of quality dimensions (Chapters 3, 4, and 6). A novel non-public data source is used for the state snapshot logs in Chapter 7. The weight estimator and Toothpaste Miner discovery techniques are are also benchmarked against comparable public alternatives, where they exist, using real-life and public event logs. For discovery algorithms, this corresponds to maturity level DM2.

C4 Formalised. Formal definitions are provided for weight estimators (Chapter 3), the Toothpaste Miner (Chapter 4), trace probability solutions (Chapters 5 and 6), metrics used in dimensions experiments (Chapter 6) and the State Snapshot Miner (Chapter 7). New data structures underlying these, including PPTs, State Snapshot Logs, and Place-Labelled Petri Nets, are also described formally. This allows the exploration of formal properties for the Toothpaste Miner in Chapter 4, such as determinism, or whether quality metrics are maintained under transformation, corresponding at least partly to discovery algorithm maturity level DM3. Solutions which composed existing techniques, such as the Stochastic Evolutionary Tree Miner (SETM) (Chapter 6) or the Alignment Estimator (Chapter 3) were not restated in formal detail, as good exact descriptions already existed in the literature.

C5 Input General. Discovery and conformance techniques in this thesis generally favour returning timely results on a wide variety of inputs versus supporting only limited inputs. Many weight estimators (Chapter 3) estimate in linear time, have no input model limitations, and can be swapped out with alternative control-flow algorithms for model inputs. Toothpaste Miner

(Chapter 4) and State Snapshot Miner (Chapter 7) terminate with a complex model rather than no model. Stochastic Play-out (Chapter 6) works with any SLPN input to approximate a stochastic language. This is a complementary approach to other existing techniques which have stricter input restrictions. For instance some discovery techniques may not terminate [130], some conformance metrics require SDFA input models [14, 101], and some conformance metrics are hard to use for cross-domain comparison [59]. In Chapter 6, we were able to extend Entropic Relevance [14] for non-SDFA inputs, and provide a broader alternative to alpha precision [59]. Input generality also facilitated the construction of the large dataset of models and metrics used for the experiments in that chapter.

C6 Tractable. Five of the weight estimators (Chapter 3), the State Snapshot Miner (Chapter 7) and Stochastic Play-out (Chapter 6) execute in linear time or a product of linear inputs. The Toothpaste Miner batch and incremental algorithms (Chapter 4) are polynomial, and executed in practical times on realistic logs. The Alignment Estimator in Chapter 3 depends on an exponential function for alignment calculation. The Trace Probability calculation on PPTs in Chapter 5 is also exponential in the worst case. Competing solutions for these latter problems use solutions such as Linear Programming, and present similar trade-offs between worst-case polynomial complexity solutions and solutions that are exponential in the worst case, but may execute quicker in practice for a number of important cases.

8.3 Limitations and Open Issues

In this section, we discuss limitations and open issues for the research introduced in this thesis. Limitations were discussed throughout the thesis, and we summarise them here.

- *Silence, duplicate labels and concurrency handling in non-alignment weight estimators.* For the weight estimators introduced in Chapter 3, five of the estimators depend on directly using activity frequencies or direct-follow frequencies, together with the structure of the input control-flow net. This is computationally cheap (linear time) but does not deal well with silent transitions or concurrency. Duplicate labels will also result in overestimates of probability for the corresponding net transitions.

- *Loop weights and duplicate labels in estimators and state-snapshot miner.* Use of direct-follow frequencies without loop detection in the non-alignment estimators (Chapter 3) and the state-snapshot miner (Chapter 7) results in them overweighting the probability of activities in loops.

- *Time complexity of the Alignment Estimator.* The alignment estimator (Chapter 3) trades off computational complexity for more sophisticated fitting of traces and models. Alignment calculation is an NP-hard problem and performance is impacted accordingly, or improved by optimisations which trade off accuracy for performance.

- *Model complexity and hidden grandchildren in Toothpaste mined models.* The Toothpaste Miner in Chapter 4 achieves polynomial time complexity by doing a form of recursive pattern-matching local to subtrees. Similar subtrees which may be simplified if adjacent to one another cannot be identified when separated by two or more levels of parent tree. In happy path cases, the intervening nodes are simplified by other rules. In less happy cases, the redundancy remains, increasing the complexity of the output model.

- *Confluence properties of Toothpaste reduction rules.* Confluence [27, p10] is a useful property for term and subtree-rewriting systems. Establishing formally whether the rules in Chapter 4 are confluent would aid in reuse in other discovery algorithms and other formal contexts.

- *Time complexity of PPT trace probability.* In the worst case, the trace probability calculation for PPTs in Chapter 5 is exponential. Some optimisations that avoid the worst case for some types of subtrees are presented with that solution. Other solutions to the TRACE-PROB problem are also super-polynomial. The time complexity of the TRACE-PROB problem is an open research question.

- *Empirically derived quality dimensions validation.* The proposed quality dimensions in Chapter 6 are empirically derived, so are particularly dependent on being strengthened or weakened by further empirical investigation and application. This would be most useful with new discovery algorithms and quality metrics. Models discovered from synthetic data would help characterise dimensions more completely. Alternative formulations of the dimensions in formal analytic terms would also complement the empirical basis and allow more general reasoning over classes of logs and models. This could parallel work on formal criteria for control-flow dimensions [9].

Finally, though all active research fields suffer some version of this problem, new metrics and algorithms have been proposed by others during the course of this thesis. Experimental evaluations performed against benchmarks available at the time may now be improved to include these new techniques and measurements.

8.4 Future Work

We have introduced a number of new discovery algorithms in this work, and as we have argued above, all of them advance the field in some distinctive way. All of them also offer some potential for extension. The weight estimation framework is a way to understand and compare any discovery technique that first employs a control-flow model. A recent algorithm for stochastic Data Petri Nets follows this structure [112]; others could too. The Toothpaste Miner framework could support new transformation rules and algorithms, particularly those looking to maintain some property under transformation. Toothpaste Miner techniques might also combine with other discovery techniques, as a preceding step, such as to turning an event log into a tree with manageable information loss, before some different computation. The idea, in Toothpaste Miner, of maintaining stochastic information throughout the discovery computation, might also apply to adaptations of successful control-flow discovery algorithms. For example, the Split Miner [15] constructs a form of internal stochastic model to make decisions about noise reduction, only to remove it before emitting the output model. The problem of mining sequential concurrent states, and using Petri net places to model them, as in the State Snapshot Miner, also deserves more research. Even without direct extension, these solutions establish benchmarks to beat: science can progress by competition as well.

As well as answering the practical question of *how likely is this sequence of activities in this process*, solutions to trace probability are foundational to a number of conformance metrics for stochastic process models, as many depend on operations performed over a stochastic language. We speculate that use of stochastic languages, or related structures, will apply to conformance metrics yet to be defined, or concepts translated from statistics or other parts of data science. Trace probability solutions may also simplify the formal demonstration of quality properties for stochastic process discovery algorithms.

Concepts like quality dimensions, including those proposed in this thesis, are strengthened, or evolved, by application to real world scenarios. This is especially true of empirically-derived dimensions such as Adhesion, Relevance, and Simplicity. We look forward to these dimensions being challenged with further empirical tests, and future formal conceptualisation of the underlying relationships.

Having made novel use of process mining on a historical data set, we hope other researchers will also explore this new domain. For the CGED-Q database of Qing civil service records specifically, a number of process mining topics parallel historical questions of interest. For example, clique analysis might be used to look at influential subgroups in a population, or concept drift applied over multi-decade timescales.

In closing, consider Figure 8.1, a photo of the Hall of Literary Profundity (文淵閣) in Beijing. It's in what is now called the Palace Museum, or more commonly in English, the Forbidden City. Elite officials, such as the ones studied in Chapter 7, were once granted membership of the hall in recognition of their accomplished literary style, an honour tracked as a title in official civil service records. It was perhaps the most exclusive library card on earth. We can imagine officials coming out of the building now, on different paths: some fresh faced graduates, newly appointed, some composing a poem, some scheming after their next promotion, some assigned to supervise a regional examination. They walk down the staircase, and across the courtyard, bound for distant provinces.

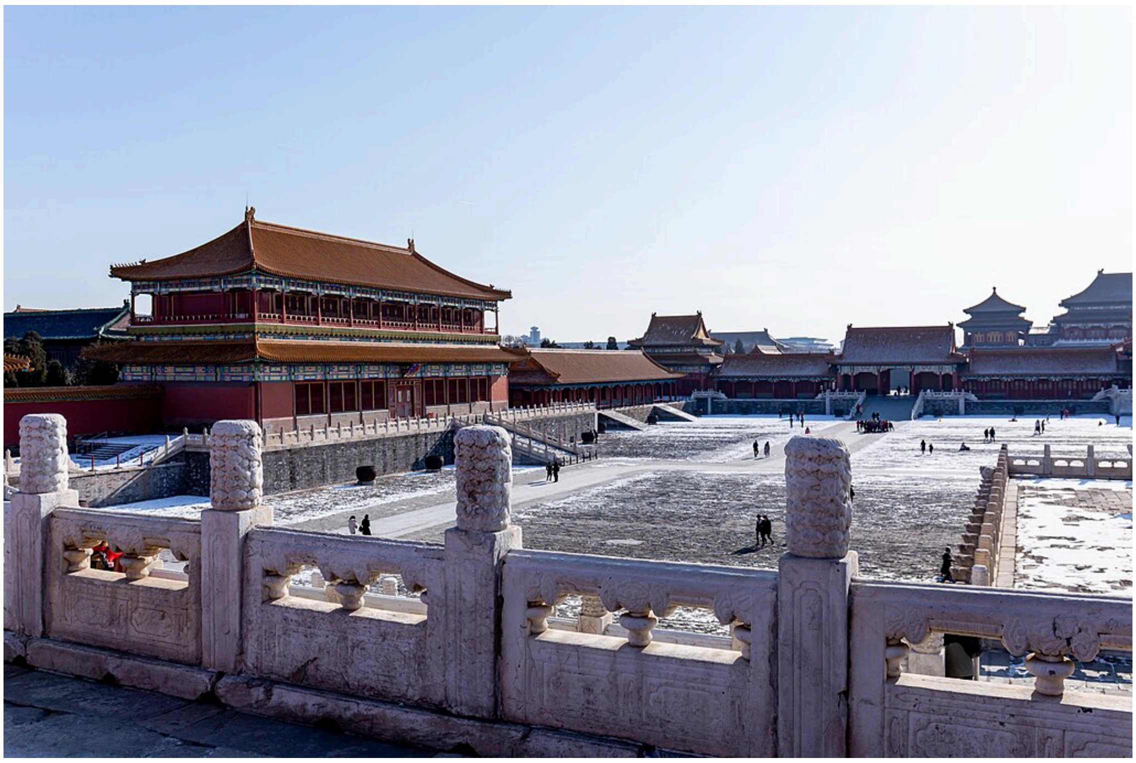


Figure 8.1: The Hall of Literary Profundity, (文淵閣) in Beijing. It is the two-storey building on the left. Photograph by Balon Greyjoy [76].

Appendix A

Tools

This appendix gives an overview of tools developed in the course of this thesis. All source code is under open source licenses and links to public repositories are provided.

A.1 Stochastic Process Discovery Miners

A.1.1 Estimators For Stochastic Discovery

Source	`https://github.com/adamburkegh/spd_we`
Language	Java
Tool requirements	JDK 1.8

This implements the six weight estimators for stochastic process discovery described in Chapter 3. They are available as a ProM plugin, or can be invoked directly using an API. XES input logs and Petri net input models are supported. SLPN output can be exported to PNML. Java 8 is used for ProM plugin support only and the code is compatible with later releases.

A.1.2 Toothpaste Miner

Source	`https://github.com/adamburkegh/toothpaste`
Language	Haskell with optional Python wrapper
Tool requirements	Stack 2.9.1 with GHC 8.10.7. Python 3.8.

This implements the Toothpaste Miners stochastic process discovery algorithms described in Chapter 4. Python wrapper scripts using pm4py are provided, which together with a Windows executable, allow the miner to be invoked without installing a Haskell development environment. XES log input and PNML output are supported.

A.1.3 Stochastic Evolutionary Tree Miner (SETM)

Source	`https://github.com/adamburkegh/spm_dim` particularly the `qut.pm.setm` package
Language	Java
Tool requirements	Java 11 or later

Implements the SETM, a genetic miner for stochastic process models described in Section 6.3.3 of Chapter 6. The SETM produces PPT models of medium quality and is suitable for laboratory exploration of possible models. It extends the Evolutionary Tree Miner [34] and makes use of the watchmaker project for genetic algorithms [65].

A.1.4 State Snapshot Miner

Source	`https://github.com/adamburkegh/statesnap-miner`
Language	Python
Tool requirements	Python 3.11 or later

This implements the State Snapshot miner for stochastic process discovery described in Chapter 7. It also includes support for reading and exporting state snapshot logs as CSV files, export of models to PNML and to pm4py objects, and visualisation of models using Graphviz DOT format.

A.2 Conformance Tools

A.2.1 PPT Trace Probability

Source	`https://github.com/adamburkegh/toothpaste`
Language	Haskell
Tool requirements	Stack 2.9.1 with GHC 8.10.7

A prototype implementation of the trace probability calculation on Probabilistic Process Trees (PPTs) described in Chapter 5 is available in Haskell and from the command line. The key module is `Toothpaste.TPConform`.

A.2.2 Conformance Metrics

A.2.3 Quality Dimension Metrics

Source	`https://github.com/adamburkegh/spm_dim` particularly the `qut.pm.spm.measures` package
Language	Java
Tool requirements	Java 11 or later

These are the implementations of the metrics in Section 6.3.5 of Chapter 6, unless an external tool is specified (e.g. earth movers distance [95]). In these cases this code wraps the other implementation. Implementations of the Adhesion, Relevance and Simplicity Dimensional Realist metrics are also provided.

A.3 Log Generation

A.3.1 Stochastic Playout

Source	`https://github.com/adamburkegh/spm_dim`
	particularly the `qut.pm.spm.playout` package
Language	Java
Tool requirements	Java 11 or later

The implementation of the log playout mechanism for stochastic process models described in Section 6.3.2 of Chapter 6. The key class is `qut.pm.spm.playout.StochasticPlayoutGenerator`.

A.4 Other Tools

A.4.1 Petri Net Fragments

Source	`https://github.com/adamburkegh/prom-helpers`
	particularly the `qut.pm.prom.helpers.PetriNetFragmentParser` class
Language	Java
Tool requirements	Java 8 or later
Source	`https://github.com/AdamBanham/koalas`
	particularly the `pmkoalas.models.pnfrag` module
Language	Python
Tool requirements	Python 3.9

This library allows the creation of Petri nets with short sketches of a few lines in human readable text that resembles Petri net diagrams. It is particularly useful making easily readable unit and functional tests. Implementations are available for Java and Python.

A.4.2 Delimited Text Log Parser

Source	`https://github.com/adamburkegh/prom-helpers`
	particularly the `qut.pm.xes.helpers.DelimitedTraceToXESConverter` class
Language	Java
Tool requirements	Java 8 or later
Source	`https://github.com/AdamBanham/koalas`
	particularly the `pmkoalas.dtlog` module
Language	Python
Tool requirements	Python 3.9

Delimited text logs allows the specification of simple logs in a text format which parallels traces in formal process mining work, e.g. the trace $\langle a, b, c \rangle$ simply becomes the line `a b c`. They are particularly useful for making easily readable unit and functional tests. Implementations are available for Java and Python.

A.4.3 Chernoff Faces

Source	`https://github.com/adamburkegh/chernoff`
Language	Python
Tool requirements	Python 3.8

Chernoff faces [49] are a whimsical data visualization technique that exploits the ability of human brains to recognize small differences in facial features easily. This project generates simple cartoon-like faces for three quality dimensions, such as those described for stochastic process models in Chapter 6.

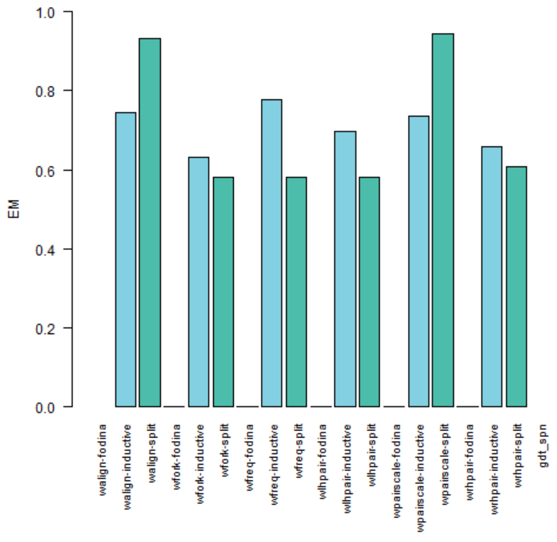
Appendix B

Estimator Detailed Results

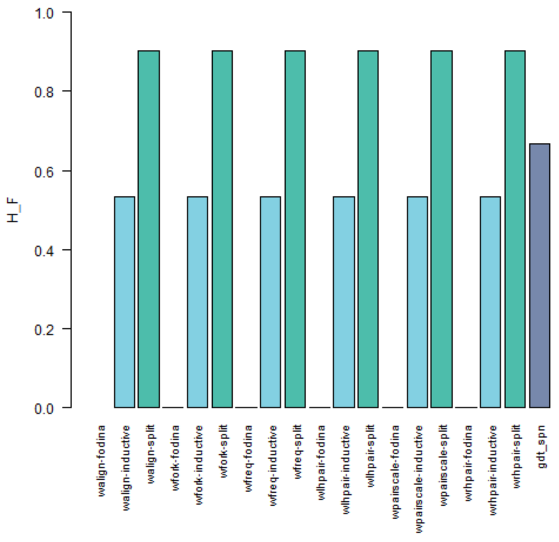
This appendix provides more detailed experimental results on an estimator framework for automatic discovery of stochastic process models, detailed in Chapter 3.

The results used to generate these figures are also available as XML data files at `https://github.com/adamburkegh/spd_we`. The result data is more detailed than the data presented here. For example, it includes the text of error messages. The source code for the estimator implementations, test run harnesses and report generation code is available at the same site.

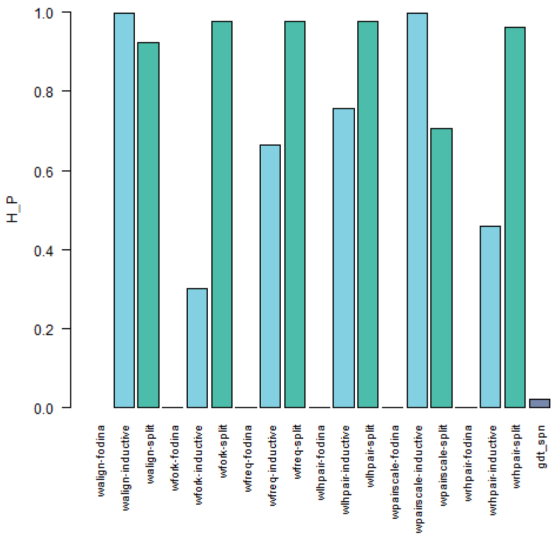
Table B.1 explains short identifiers for miners and miner-estimator combinations used as keys throughout this appendix. Results are categorized by {*estimator*}-{*control flow algorithm*}, plus GDT_SPN discovery. Figures B.1-B.7 show the results by log across the three quality measures used.



(a) earth movers' EM

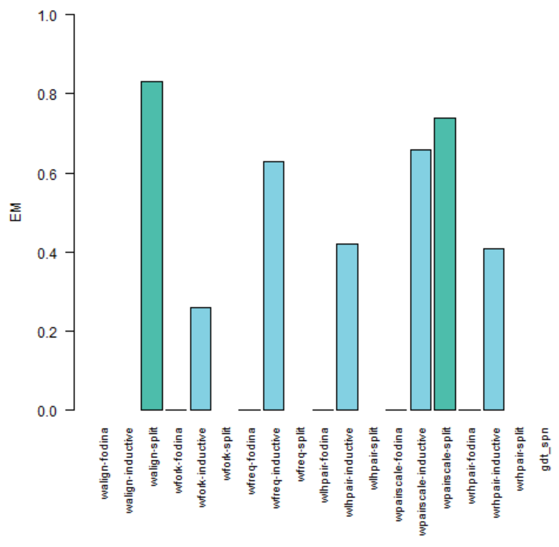


(b) entropy-recall H_F

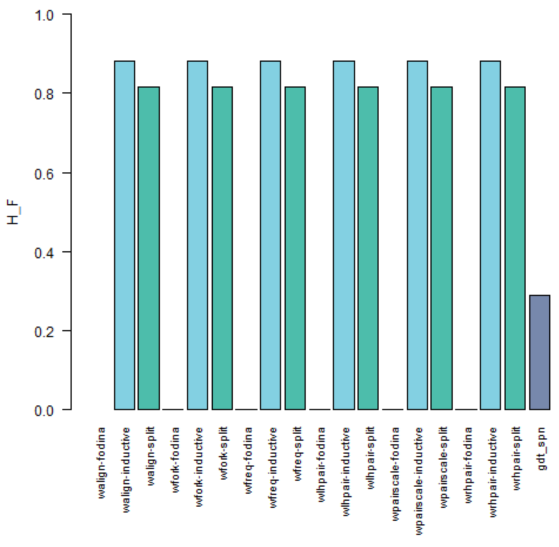


(c) entropy-precision H_P

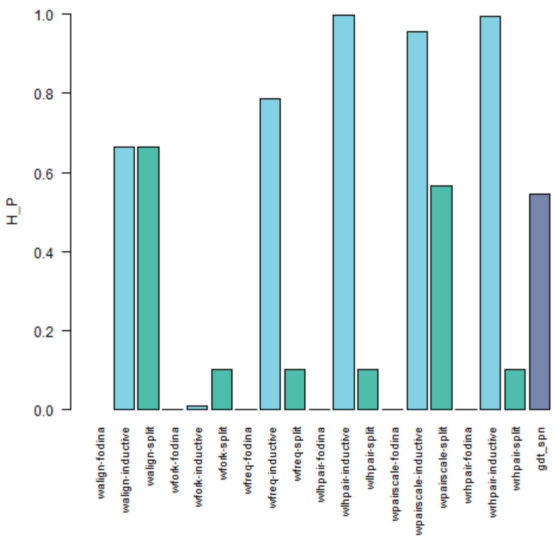
Figure B.1: Results on BPIC 2013 closed log.



(a) earth movers' EM

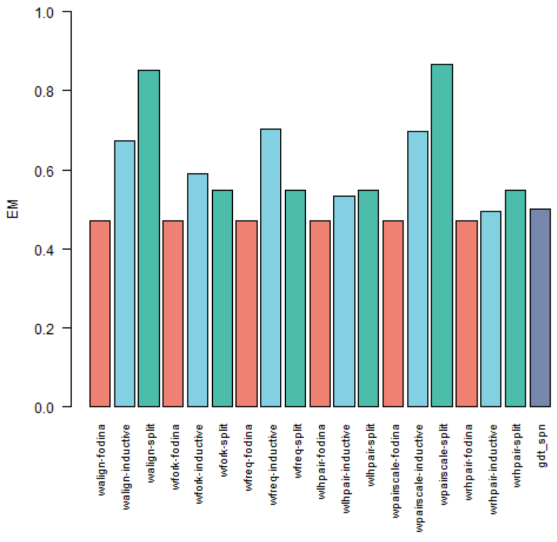


(b) entropy-recall H_F

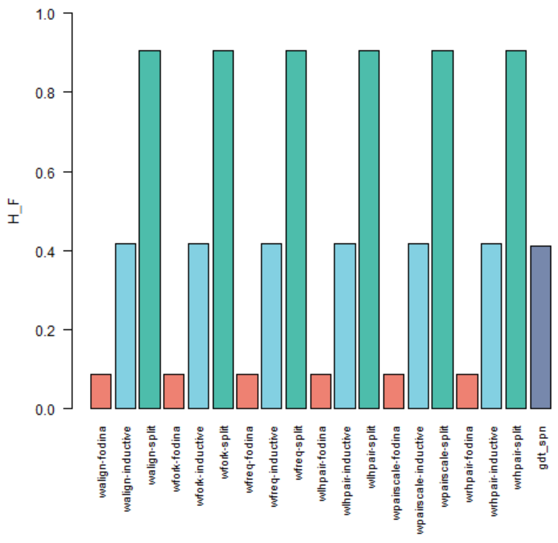


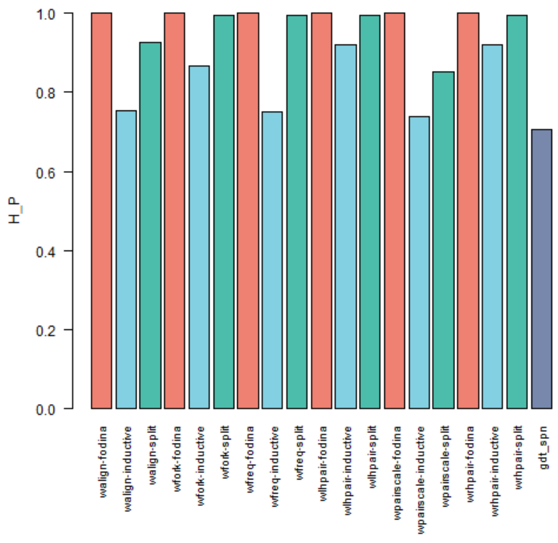
(c) entropy-precision H_P

Figure B.2: Results on BPIC 2013 incidents log.



(a) earth movers' EM

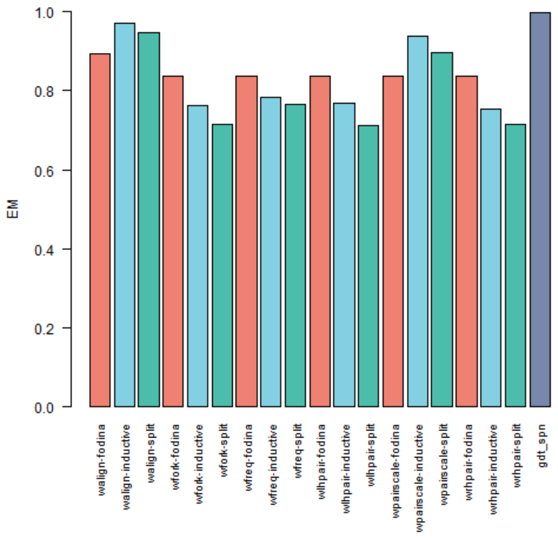




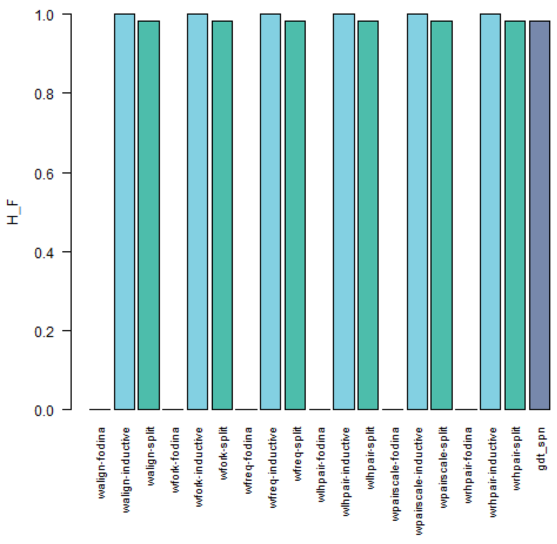
(b) entropy-recall H_F

(c) entropy-precision H_P

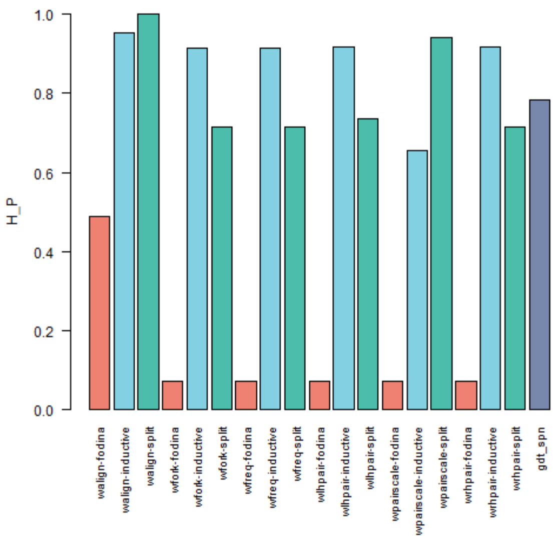
Figure B.3: Results on BPIC 2013 open log.



(a) earth movers' EM

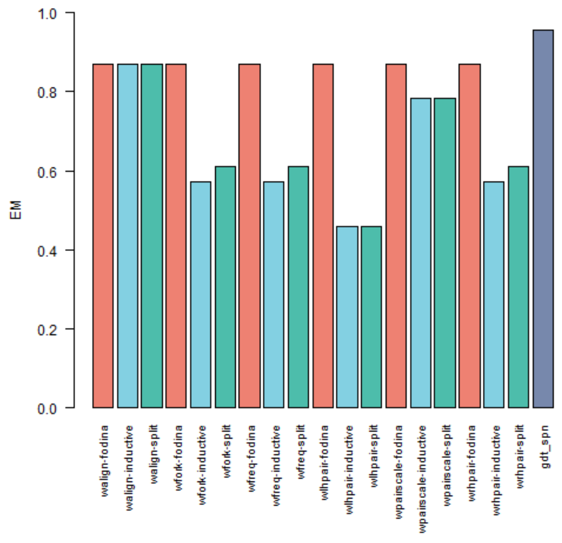


(b) entropy-recall H_F

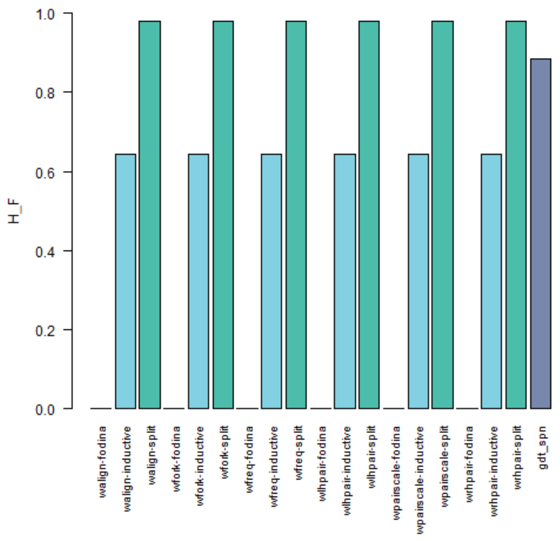


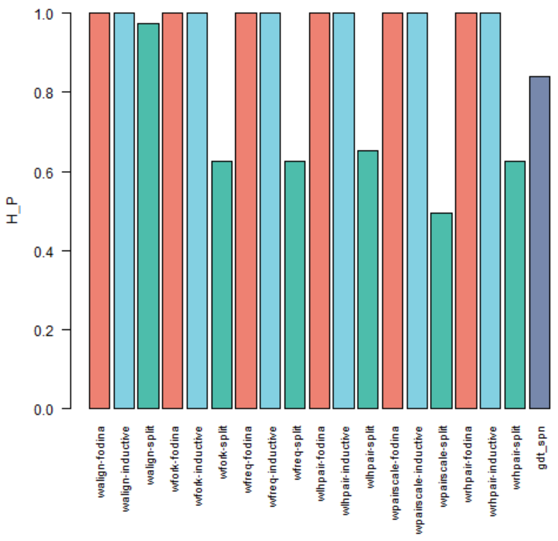
(c) entropy-precision H_P

Figure B.4: Results on BPIC 2018 Control log.



(a) earth movers' EM

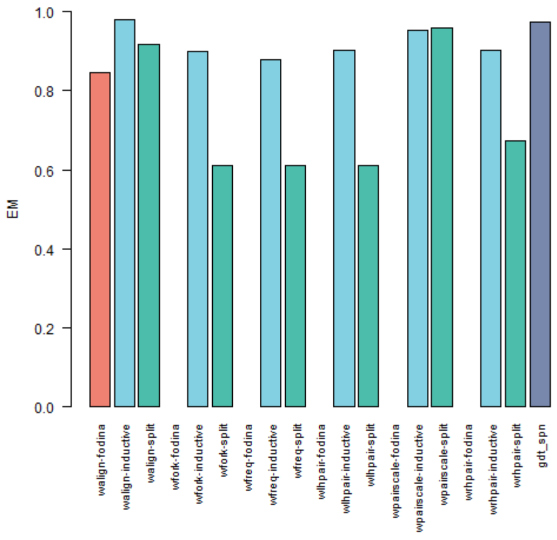




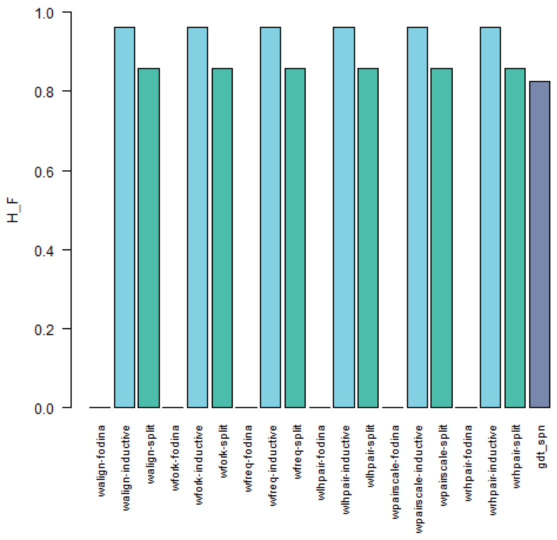
(b) entropy-recall H_F

(c) entropy-precision H_P

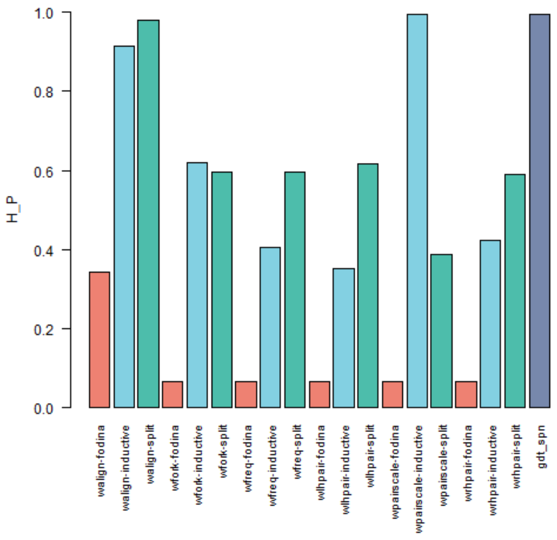
Figure B.5: Results on BPIC 2018 Dept log.



(a) earth movers' EM

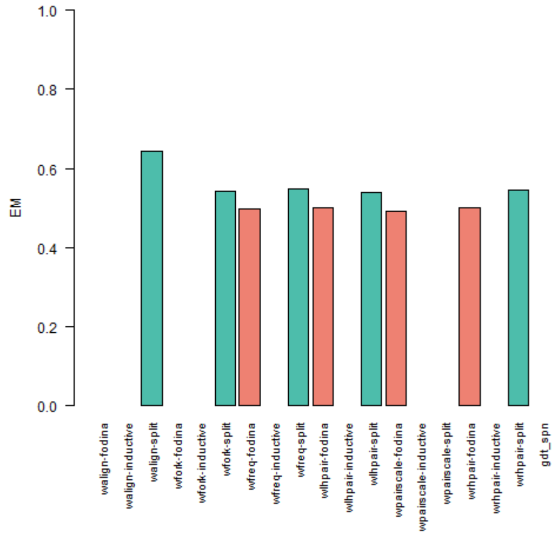


(b) entropy-recall H_F

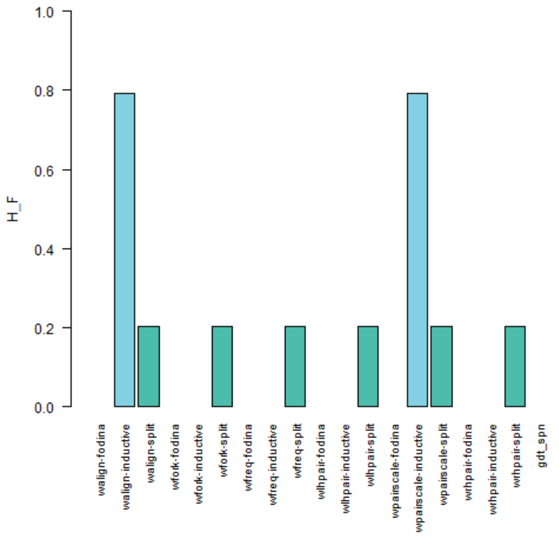


(c) entropy-precision H_P

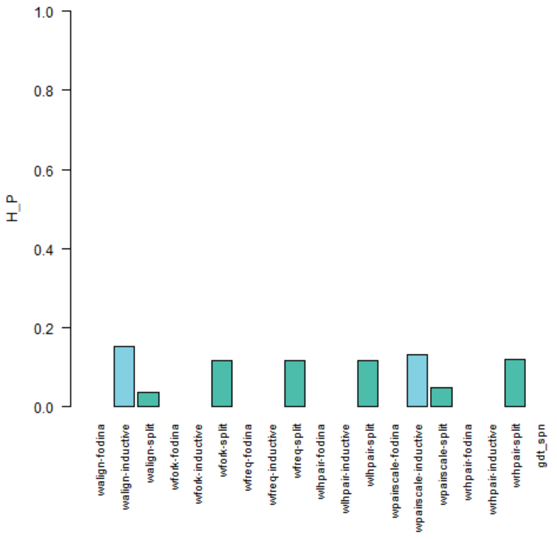
Figure B.6: Results on BPIC 2018 Reference log.



(a) earth movers' EM



(b) entropy-recall H_F



(c) entropy-precision H_P

Figure B.7: Results on Sepsis log.

Short Id	Artifact Creator Full Name	Type
walign	Alignment	Estimator
wfork	Fork-Distributed	Estimator
wfreq	Frequency	Estimator
wlhpair	Activity Pair Left-Handed	Estimator
wpairscale	Mean-Scaled Activity Pair Right-Handed	Estimator
wrhpair	Activity Pair Right-Handed	Estimator
fodina	Fodina Miner [32]	Control Flow Miner
inductive	Inductive Miner [96]	Control Flow Miner
split	Split Miner [15]	Control Flow Miner
gdt_spn	GDT_SPN discovery [130]	Stochastic Miner

Table B.1: Discovery technique overview, with estimators and their control-flow in alphabetical order by short name.

Bibliography

[1] *'Sea of Steps', Wells Cathedral, Wells, Somerset.* National Monument Record: aa66/00136. 1948. URL: `https://historicengland.org.uk/services-skills/education/educational-images/sea-of-steps-wells-cathedral-wells-4275` (visited on 21/07/2023) (cit. on p. 2).

[2] ISO/IEC 15909-2:2011. *Systems and software engineering –High-level Petri nets –Part 2: Transfer format, International Standard.* Standard. International Organization for Standardization, Feb. 2011 (cit. on p. 6).

[3] Wil M. P van der Aalst, Arya Adriansyah and Boudewijn van Dongen. 'Replaying history on process models for conformance checking and performance analysis'. en. In: *WIREs Data Mining and Knowledge Discovery* 2.2 (2012), pp. 182–192. ISSN: 1942-4795. DOI: `10.1002/widm.1045`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1045` (visited on 30/08/2020) (cit. on pp. 10, 25, 32).

[4] Wil M. P. van der Aalst, M. Pesic and H. Schonenberg. 'Declarative workflows: Balancing between flexibility and support'. en. In: *Computer Science - Research and Development* 23.2 (May 2009), pp. 99–113. ISSN: 0949-2925. DOI: `10.1007/s00450-009-0057-9`. URL: `https://doi.org/10.1007/s00450-009-0057-9` (visited on 18/06/2020) (cit. on p. 7).

[5] Wil M.P. van der Aalst. 'Academic View: Development of the Process Mining Discipline'. en. In: *Process Mining in Action: Principles, Use Cases and Outlook.* Ed. by Lars Reinkemeyer. Cham: Springer International Publishing, 2020, pp. 181–196. ISBN: 978-3-030-40172-6. DOI: `10.1007/978-3-030-40172-6_21`. URL: `https://doi.org/10.1007/978-3-030-40172-6_21` (visited on 20/03/2020) (cit. on p. 3).

[6] Wil M.P. van der Aalst. *Process Mining: Data Science in Action.* en. 2nd ed. Berlin Heidelberg: Springer-Verlag, 2016. ISBN: 978-3-662-49850-7. DOI: `10.1007/978-3-662-49851-4`. URL: `https://www.springer.com/gp/book/9783662498507` (visited on 17/02/2020) (cit. on pp. 1, 3–11, 24, 25, 27, 32, 42, 61, 85, 90, 103–106, 110, 120, 124, 129, 131, 145, 148).

[7] Wil M.P. van der Aalst. *Teleclaims reference log.* 2011. DOI: `10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`. URL: `https://www.processmining.org/old-version/files/chapter_8.zip` (visited on 04/08/2023) (cit. on pp. 16, 41, 82).

[8] Wil M.P. van der Aalst and A. J. M. M. Weijters. ‘Process mining: a research agenda’. en. In: *Computers in Industry*. Process / Workflow Mining 53.3 (Apr. 2004), pp. 231–244. ISSN: 0166-3615. DOI: 10.1016/j.compind.2003.10.001. URL: https://www.sciencedirect.com/science/article/pii/S0166361503001945 (visited on 03/12/2021) (cit. on p. 86).

[9] Wil MP van der Aalst. ‘Relating Process Models and Event Logs-21 Conformance Propositions.’ In: *ATAED@ Petri Nets/ACSD*. 2018, pp. 56–74 (cit. on pp. 108, 109, 115, 151).

[10] Andrew Abbott and Alexandra Hrycak. ‘Measuring resemblance in sequence data: An optimal matching analysis of musicians’ careers’. In: *American journal of sociology* 96.1 (1990). Publisher: University of Chicago Press, pp. 144–185 (cit. on pp. 11, 132).

[11] M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte and A. Cumani. ‘The effect of execution policies on the semantics and analysis of stochastic Petri nets’. In: *IEEE Transactions on Software Engineering* 15.7 (July 1989). Conference Name: IEEE Transactions on Software Engineering, pp. 832–846. ISSN: 1939-3520. DOI: 10.1109/32.29483 (cit. on p. 22).

[12] A Alharbi, A Bulpitt and OA Johnson. ‘Towards Unsupervised Detection of Process Models in Healthcare.’ English. In: *Studies in Health Technology and Informatics* 247 (Jan. 2018), pp. 381–385. ISSN: 0926-9630, 1879-8365. URL: https://europepmc.org/article/med/29677987 (visited on 05/09/2020) (cit. on pp. 5, 12).

[13] Amirah Mohammed Alharbi. ‘Unsupervised Abstraction for Reducing the Complexity of Healthcare Process Models’. phd. University of Leeds, July 2019. URL: http://etheses.whiterose.ac.uk/25103/ (visited on 10/03/2020) (cit. on pp. 5, 12).

[14] Hanan Alkhammash, Artem Polyvyanyy, Alistair Moffat and Luciano García-Bañuelos. ‘Entropic relevance: A mechanism for measuring stochastic process models discovered from event data’. en. In: *Information Systems* 107 (Nov. 2022), p. 101922. ISSN: 0306-4379. DOI: 10.1016/j.is.2021.101922. URL: https://www.sciencedirect.com/science/article/pii/S0306437921001277 (visited on 10/01/2022) (cit. on pp. 7, 12, 28, 43, 55, 90, 101, 104, 106, 109, 110, 120, 150).

[15] Adriano Augusto, Raffaele Conforti, Marlon Dumas, Marcello La Rosa and Artem Polyvyanyy. ‘Split miner: automated discovery of accurate and simple business process models from event logs’. en. In: *Knowledge and Information Systems* 59.2 (May 2019), pp. 251–284. ISSN: 0219-3116. DOI: 10.1007/s10115-018-1214-x. URL: https://doi.org/10.1007/s10115-018-1214-x (visited on 30/08/2020) (cit. on pp. 34, 151, 167).

[16] Charles Babbage. *On the economy of machinery and manufactures*. John Murray, 1832 (cit. on p. 89).

[17] Alan Baker. ‘Simplicity’. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2016. Metaphysics Research Lab, Stanford University, 2016. URL: https://plato.stanford.edu/archives/win2016/entries/simplicity/ (cit. on pp. 9, 10).

[18] Alexander T. J. Barron, Jenny Huang, Rebecca L. Spang and Simon DeDeo. 'Individuals, institutions, and innovation in the debates of the French Revolution'. en. In: *Proceedings of the National Academy of Sciences* 115.18 (May 2018). Publisher: National Academy of Sciences Section: Social Sciences, pp. 4607–4612. ISSN: 0027-8424, 1091-6490. DOI: `10.1073/pnas.1717729115`. URL: `https://www.pnas.org/content/115/18/4607` (visited on 27/04/2021) (cit. on p. 131).

[19] Leonard E Baum, Ted Petrie, George Soules and Norman Weiss. 'A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains'. In: *The annals of mathematical statistics* 41.1 (1970). Publisher: JSTOR, pp. 164–171 (cit. on p. 6).

[20] F. Bause and P.S. Kritzinger. *Stochastic Petri Nets: An Introduction to the Theory*. Vieweg+Teubner Verlag, 2002. ISBN: 978-3-528-15535-3 (cit. on pp. 4, 11, 21, 22, 90).

[21] Elena Bellodi, Fabrizio Riguzzi and Evelina Lamma. 'Probabilistic Declarative Process Mining'. en. In: *Knowledge Science, Engineering and Management*. Ed. by Yaxin Bi and Mary-Anne Williams. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 292–303. ISBN: 978-3-642-15280-1. DOI: `10.1007/978-3-642-15280-1_28` (cit. on pp. 7, 11, 132).

[22] Elena Bellodi, Fabrizio Riguzzi and Evelina Lamma. 'Statistical relational learning for workflow mining'. en. In: *Intelligent Data Analysis* 20.3 (Jan. 2016). Publisher: IOS Press, pp. 515–541. ISSN: 1088-467X. DOI: `10.3233/IDA-160818`. URL: `https://content.iospress.com/articles/intelligent-data-analysis/ida818` (visited on 11/11/2020) (cit. on p. 7).

[23] Sara Belluccini, Rocco De Nicola, Barbara Re and Francesco Tiezzi. 'PALM: A Technique for Process ALgebraic Specification Mining'. en. In: *Integrated Formal Methods*. Ed. by Brijesh Dongol and Elena Troubitsyna. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 397–418. ISBN: 978-3-030-63461-2. DOI: `10.1007/978-3-030-63461-2_22` (cit. on p. 86).

[24] Giacomo Bergami, Fabrizio Maria Maggi, Marco Montali and Rafael Peñaloza. 'Probabilistic trace alignment'. In: *2021 3rd International Conference on Process Mining (ICPM)*. IEEE, 2021, pp. 9–16 (cit. on pp. 8, 91).

[25] Simona Bernardi, José Ignacio Requeno, Christophe Joubert and Alberto Romeu. 'A systematic approach for performance evaluation using process mining: the POSIDONIA operations case study'. In: *Proceedings of the 2nd International Workshop on Quality-Aware DevOps*. QUDOS 2016. Saarbrücken, Germany: Association for Computing Machinery, July 2016, pp. 24–29. ISBN: 978-1-4503-4411-1. DOI: `10.1145/2945408.2945413`. URL: `http://doi.org/10.1145/2945408.2945413` (visited on 19/02/2020) (cit. on p. 10).

[26] Marco Bernardo and Roberto Gorrieri. 'A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time'. en. In: *Theoretical Computer Science* 202.1 (July 1998), pp. 1–54. ISSN: 0304-3975. DOI: `10.1016/S0304-3975(97)00127-8`. URL: `https://www.sciencedirect.com/science/article/pii/S0304397597001278` (visited on 03/12/2021) (cit. on p. 86).

[27] M. Bezem, J.W. Klop, E. Barendsen, R. de Vrijer and Terese. *Term Rewriting Systems.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003. ISBN: 978-0-521-39115-3. URL: `https://books.google.com.au/books?id=7QQ5u-4tRUkC` (cit. on pp. 58, 87, 151).

[28] E. Bogdanov, I. Cohen and A. Gal. 'Conformance Checking over Stochastically Known Logs'. English. In: *Lecture Notes in Business Information Processing* 458 LNBIP (2022). ISBN: 9783031161704, pp. 105–119. ISSN: 1865-1348. DOI: `10.1007/978-3-031-16171-1_7` (cit. on pp. 10, 11).

[29] Dominic Breuker, Martin Matzner, Patrick Delfmann and Jörg Becker. 'Comprehensible Predictive Models for Business Processes.' In: *MIS Q.* 40.4 (2016), pp. 1009–1034 (cit. on p. 6).

[30] T. Brockhoff, M. S. Uysal and W. M. P. van der Aalst. 'Time-aware Concept Drift Detection Using the Earth Mover' s Distance'. In: *2020 2nd International Conference on Process Mining (ICPM).* Oct. 2020, pp. 33–40. DOI: `10.1109/ICPM49681.2020.00016` (cit. on p. 127).

[31] S. K. L. M. vanden Broucke, J. De Weerdt, J. Vanthienen and B. Baesens. 'Determining Process Model Precision and Generalization with Weighted Artificial Negative Events'. In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (Aug. 2014). Conference Name: IEEE Transactions on Knowledge and Data Engineering, pp. 1877–1889. ISSN: 1558-2191. DOI: `10.1109/TKDE.2013.130` (cit. on p. 10).

[32] Seppe K. L. M. vanden Broucke and Jochen De Weerdt. 'Fodina: A robust and flexible heuristic process discovery technique'. en. In: *Decision Support Systems.* Smart Business Process Management 100 (Aug. 2017), pp. 109–118. ISSN: 0167-9236. DOI: `10.1016/j.dss.2017.04.005`. URL: `http://www.sciencedirect.com/science/article/pii/S0167923617300647` (visited on 14/08/2020) (cit. on pp. 34, 167).

[33] J. C. a. M. Buijs, Boudewijn F. van Dongen and Wil M.P. van der Aalst. 'Quality Dimensions in Process Discovery: The Importance of Fitness, Precision, Generalization and Simplicity'. In: *International Journal of Cooperative Information Systems* 23.01 (Mar. 2014). Publisher: World Scientific Publishing Co., p. 1440001. ISSN: 0218-8430. DOI: `10.1142/S0218843014400012`. URL: `https://www.worldscientific.com/doi/abs/10.1142/S0218843014400012` (visited on 23/04/2021) (cit. on p. 105).

[34] J. C. A. M. Buijs, Boudewijn. F. van Dongen and Wil M.P. van der Aalst. 'A genetic algorithm for discovering process trees'. In: *2012 IEEE Congress on Evolutionary Computation.* ISSN: 1941-0026. June 2012, pp. 1–8. DOI: `10.1109/CEC.2012.6256458` (cit. on pp. 42, 105, 112, 156).

[35] Joos C. A. M. Buijs, Boudewijn F. van Dongen and Wil M. P. van der Aalst. 'On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery'. en. In: *On the Move to Meaningful Internet Systems: OTM 2012.* Ed. by Robert Meersman, Hervé Panetto, Tharam Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi and Isabel F. Cruz. Lecture Notes in Computer Science. Berlin,

Heidelberg: Springer, 2012, pp. 305–322. ISBN: 978-3-642-33606-5. DOI: `10.1007/978-3-642-33606-5_19` (cit. on pp. 8, 10).

[36] Adam T. Burke, Sander J.J. Leemans, Moe T. Wynn, Wil M.P van der Aalst and Arthur H.M. ter Hofstede. 'Stochastic Process Model-Log Quality Dimensions: An Experimental Study'. In: *2022 4th International Conference on Process Mining (ICPM)*. Oct. 2022, pp. 80–87. DOI: `10.1109/ICPM57379.2022.9980707` (cit. on p. 120).

[37] Manuel Camargo, Daniel Báron, Marlon Dumas and Oscar González-Rojas. 'Learning business process simulation models: A Hybrid process mining and deep learning approach'. en. In: *Information Systems* (June 2023), p. 102248. ISSN: 0306-4379. DOI: `10.1016/j.is.2023.102248`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437923000844` (visited on 05/07/2023) (cit. on p. 6).

[38] Manuel Camargo, Marlon Dumas and Oscar González-Rojas. 'Automated discovery of business process simulation models from event logs'. en. In: *Decision Support Systems* 134 (July 2020), p. 113284. ISSN: 0167-9236. DOI: `10.1016/j.dss.2020.113284`. URL: `http://www.sciencedirect.com/science/article/pii/S0167923620300397` (visited on 12/10/2020) (cit. on pp. 6, 12, 28, 42).

[39] Cameron Campbell, Bijia Chen et al. 'Nominative Linkage of Records of Officials in the China Government Employee Dataset-Qing (CGED-Q)'. In: *Historical Life Course Studies* 12 (2022), pp. 233–259 (cit. on p. 141).

[40] Cameron Campbell and James Lee. 'Historical Chinese Microdata. 40 Years of Dataset Construction by the Lee-Campbell Research Group'. en. In: *Historical Life Course Studies* (Sept. 2020). URL: `/article/historical-chinese-microdata-40-years-dataset-construction-lee-campbell-research-group` (visited on 26/10/2020) (cit. on pp. 15, 130, 131).

[41] Josep Carmona, J. Cortadella and M. Kishinevsky. 'New Region-Based Algorithms for Deriving Bounded Petri Nets'. In: *IEEE Transactions on Computers* 59.3 (Mar. 2010). Conference Name: IEEE Transactions on Computers, pp. 371–384. ISSN: 1557-9956. DOI: `10.1109/TC.2009.131` (cit. on p. 42).

[42] Josep Carmona, Jordi Cortadella and Michael Kishinevsky. 'A region-based algorithm for discovering Petri nets from event logs'. In: *Business Process Management: 6th International Conference, BPM 2008, Milan, Italy, September 2-4, 2008. Proceedings 6*. Springer, 2008, pp. 358–373 (cit. on p. 42).

[43] Josep Carmona, Boudewijn F. van Dongen, Andreas Solti and Matthias Weidlich. *Conformance Checking: Relating Processes and Models*. en. Springer International Publishing, 2018. ISBN: 978-3-319-99413-0. DOI: `10.1007/978-3-319-99414-7`. URL: `https://www.springer.com/gp/book/9783319994130` (visited on 20/05/2020) (cit. on pp. 7, 9).

[44] Rafael C. Carrasco and Jose Oncina. 'Learning stochastic regular grammars by means of a state merging method'. en. In: *Grammatical Inference and Applications*. Ed. by Rafael C. Carrasco and Jose Oncina. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1994, pp. 139–152. ISBN: 978-3-540-48985-6. DOI: 10.1007/3-540-58473-0_144 (cit. on pp. 5, 12, 42).

[45] Berny Carrera and Jae-Yoon Jung. 'Constructing probabilistic process models based on hidden markov models for resource allocation'. In: *International Conference on Business Process Management*. Springer, 2014, pp. 477–488 (cit. on p. 5).

[46] Bijia Chen. 'Origins and Career Patterns of the Qing Government Officials (1850-1912): Evidence from the China Government Employee Dataset-Qing (CGED-Q)'. English. PhD Thesis. Hong Kong University of Science and Technology, 2019 (cit. on pp. 131, 138, 140).

[47] Bijia Chen, Cameron Campbell, Yuxue Ren and James Lee. 'Big Data for the Study of Qing Officialdom: The China Government Employee Database-Qing (CGED-Q)'. en. In: *Journal of Chinese History* 4.2 (July 2020). Publisher: Cambridge University Press, pp. 431–460. ISSN: 2059-1632, 2059-1640. DOI: 10.1017/jch.2020.15. URL: http://www.cambridge.org/core/journals/journal-of-chinese-history/article/big-data-for-the-study-of-qing-officialdom-the-china-government-employee-databaseqing-cgedq/08012BBACCA482B19D7FE2724FDCC840 (visited on 16/10/2020) (cit. on pp. 15, 129–131, 138).

[48] Haoxun Chen, L. Amodeo, Feng Chu and K. Labadi. 'Modeling and performance evaluation of supply chains using batch deterministic and stochastic Petri nets'. In: *IEEE Transactions on Automation Science and Engineering* 2.2 (Apr. 2005), pp. 132–144. ISSN: 1558-3783. DOI: 10.1109/TASE.2005.844537 (cit. on p. 11).

[49] Herman Chernoff. 'The use of faces to represent points in k-dimensional space graphically'. In: *Journal of the American statistical Association* 68.342 (1973). Publisher: Taylor & Francis, pp. 361–368 (cit. on p. 158).

[50] Claudio Cioffi-Revilla. 'Computational social science'. en. In: *WIREs Computational Statistics* 2.3 (2010). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.95, pp. 259–271. ISSN: 1939-0068. DOI: 10.1002/wics.95. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.95 (visited on 29/01/2023) (cit. on p. 131).

[51] Alistair Cockburn. *Agile Software Development: The Cooperative Game*. Agile Software Development Series. Pearson Education, 2006. ISBN: 978-0-321-63007-0. URL: https://books.google.com.au/books?id=i39yimbrzh4C (cit. on p. 15).

[52] I. Cohen and A. Gal. 'Uncertain process data with probabilistic knowledge: Problem characterization and challenges'. English. In: vol. 2938. ISSN: 1613-0073. 2021, pp. 51–56 (cit. on pp. 10, 11).

[53] Timothy Colburn. 'Methodology of computer science'. In: *The Blackwell guide to the philosophy of computing and information* (2004). Publisher: Wiley Online Library, pp. 318–326 (cit. on p. 15).

[54] Benjamin Cornwell. *Social sequence analysis: Methods and applications*. Vol. 37. Cambridge University Press, 2015 (cit. on pp. 11, 132).

[55] Kostadin Damevski, Hui Chen, David Shepherd and Lori Pollock. 'Interactive exploration of developer interaction traces using a hidden markov model'. In: *Proceedings of the 13th International Conference on Mining Software Repositories*. MSR '16. Austin, Texas: Association for Computing Machinery, May 2016, pp. 126–136. ISBN: 978-1-4503-4186-8. DOI: `10.1145/2901739.2901741`. URL: `http://doi.org/10.1145/2901739.2901741` (visited on 19/02/2020) (cit. on p. 5).

[56] Massimiliano De Leoni and Wil MP Van Der Aalst. 'Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming'. In: *Business Process Management: 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*. Springer, 2013, pp. 113–129 (cit. on pp. 14, 25).

[57] Morris H DeGroot and Mark J Schervish. *Probability and statistics*. 4th. Pearson Education London, UK: 2014 (cit. on p. 71).

[58] Gilles Deleuze and Felix Guattari. *A Thousand Plateaus: Capitalism and Schizophrenia*. Capitalism and schizophrenia. University of Minnesota Press, 1987. ISBN: 978-0-8166-1402-8. URL: `https://books.google.com.au/books?id=PdeOQtcOQTgC` (cit. on p. 103).

[59] Benoît Depaire, Gert Janssenswillen and Sander J. J. Leemans. 'Alpha Precision: Estimating the Significant System Behavior in a Model'. en. In: *Business Process Management Forum*. Ed. by Claudio Di Ciccio, Remco Dijkman, Adela del Río Ortega and Stefanie Rinderle-Ma. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2022, pp. 120–136. ISBN: 978-3-031-16171-1. DOI: `10.1007/978-3-031-16171-1_8` (cit. on pp. 8, 12, 28, 43, 81, 82, 90, 106, 110, 115, 150).

[60] Clive Dilnot. 'Designing In The World of the Naturalised Artificial'. In: *Design in crisis; New Worlds, Philosophies and Practices*. Ed. by Tony Fry and Adam Nocek. Taylor & Francis, 2020 (cit. on p. 41).

[61] Boudewijn van Dongen and Florian Borchert. *BPI Challenge 2018*. Dataset. https://doi.org/10.4121/uuid:3301 95e8-4ff0-98a4-901f1f204972. 2018 (cit. on pp. 16, 82).

[62] Boudewijn F. van Dongen, J. Carmona and T. Chatain. 'A Unified Approach for Measuring Precision and Generalization Based on Anti-alignments'. en. In: *Business Process Management*. Ed. by Marcello La Rosa, Peter Loos and Oscar Pastor. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 39–56. ISBN: 978-3-319-45348-4. DOI: `10.1007/978-3-319-45348-4_3` (cit. on p. 10).

[63] Boudewijn F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters and Wil M.P. van der Aalst. 'The ProM Framework: A New Era in Process Mining Tool Support'. en. In: *Applications and Theory of Petri Nets 2005*. Ed. by Gianfranco Ciardo and Philippe Darondeau. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 444–454. ISBN: 978-3-540-31559-9. DOI: `10.1007/11494744_25` (cit. on p. 33).

[64] Sebastian Dunzer, Matthias Stierle, Martin Matzner and Stephan Baier. 'Conformance checking: a state-of-the-art literature review'. In: *Proceedings of the 11th International Conference on Subject-Oriented Business Process Management*. S-BPM ONE '19. New York, NY, USA: Association for Computing Machinery, June 2019, pp. 1–10. ISBN: 978-1-4503-6250-4. DOI: `10.1145/3329007.3329014`. URL: `https://dl.acm.org/doi/10.1145/3329007.3329014` (visited on 20/07/2023) (cit. on p. 3).

[65] Daniel W Dyer. 'Evolutionary computation in java: A practical guide to the watchmaker framework'. In: *URL https://watchmaker. uncommons. org/manual/ch01s02. html* (2008) (cit. on p. 156).

[66] Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen and Lijun Zhang. 'A Semantics for Every GSPN'. en. In: *Application and Theory of Petri Nets and Concurrency*. Ed. by José-Manuel Colom and Jörg Desel. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 90–109. ISBN: 978-3-642-38697-8. DOI: `10.1007/978-3-642-38697-8_6` (cit. on pp. 5, 23, 42).

[67] Christian Eisentraut, Holger Hermanns and Lijun Zhang. 'Concurrency and composition in a stochastic world'. In: *CONCUR 2010-Concurrency Theory: 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings 21*. Springer, 2010, pp. 21–39 (cit. on p. 23).

[68] Benjamin A Elman. *Civil examinations and meritocracy in late imperial China*. Harvard University Press, 2013 (cit. on pp. 129, 138).

[69] Javier Esparza, Stefan Römer and Walter Vogler. 'An Improvement of McMillan's Unfolding Algorithm'. en. In: *Formal Methods in System Design* 20.3 (May 2002), pp. 285–310. ISSN: 1572-8102. DOI: `10.1023/A:1014746130920`. URL: `https://doi.org/10.1023/A:1014746130920` (visited on 21/02/2022) (cit. on p. 100).

[70] European Mathematical Society. 'Model selection'. In: *Encyclopedia of Mathematics*. EMS Press, Feb. 2021. URL: `http://encyclopediaofmath.org/index.php?title=Model_selection&oldid=50933` (cit. on p. 9).

[71] Catherine Forbes, Merran Evans, Nicholas Hastings and Brian Peacock. *Statistical Distributions*. en. John Wiley & Sons, Ltd, 2010. ISBN: 978-0-470-62724-2. URL: `http://onlinelibrary.wiley.com/doi/abs/10.1002/9780470627242` (visited on 19/12/2021) (cit. on p. 70).

[72] Cleiton dos Santos Garcia, Alex Meincheim, Elio Ribeiro Faria Junior, Marcelo Rosano Dallagassa, Denise Maria Vecino Sato, Deborah Ribeiro Carvalho, Eduardo Alves Portela Santos and Edson Emilio Scalabrin. 'Process mining techniques and applications – A systematic mapping study'. en. In: *Expert Systems with Applications* 133 (Nov. 2019), pp. 260–295. ISSN: 0957-4174. DOI: `10.1016/j.eswa.2019.05.003`. URL: `https://www.sciencedirect.com/science/article/pii/S0957417419303161` (visited on 19/03/2023) (cit. on p. 105).

[73] S. Geman and D. Geman. 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (Nov. 1984). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 721–741. ISSN: 1939-3539. DOI: `10.1109/TPAMI.1984.4767596` (cit. on p. 6).

[74] Stephen Gilmore, Jane Hillston and Marina Ribaudo. 'PEPA Nets: A Structured Performance Modelling Formalism'. en. In: *Computer Performance Evaluation: Modelling Techniques and Tools*. Ed. by Tony Field, Peter G. Harrison, Jeremy Bradley and Uli Harder. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2002, pp. 111–130. ISBN: 978-3-540-46029-9. DOI: `10.1007/3-540-46029-2_7` (cit. on p. 86).

[75] John C Gower. 'A general coefficient of similarity and some of its properties'. In: *Biometrics* (1971). Publisher: JSTOR, pp. 857–871 (cit. on p. 114).

[76] Balon Greyjoy. *20200110 Hall of Literary Glory*. 2020. URL: `https://commons.wikimedia.org/wiki/File:20200110_Hall_of_Literary_Glory_gate-1.jpg` (visited on 21/07/2023) (cit. on p. 153).

[77] Volker Gruhn and Ralf Laue. 'Adopting the Cognitive Complexity Measure for Business Process Models'. In: *2006 5th IEEE International Conference on Cognitive Informatics*. Vol. 1. July 2006, pp. 236–241. DOI: `10.1109/COGINF.2006.365702` (cit. on p. 33).

[78] Myron P. Gutmann, Emily Klancher Merchant and Evan Roberts. ' "Big Data" in Economic History'. en. In: *The Journal of Economic History* 78.1 (Mar. 2018). Publisher: Cambridge University Press, pp. 268–299. ISSN: 0022-0507, 1471-6372. DOI: `10.1017/S0022050718000177`. URL: `https://www.cambridge.org/core/journals/journal-of-economic-history/article/abs/big-data-in-economic-history/C6309D815663917BBF03DC6B95CE62B4` (visited on 17/07/2023) (cit. on p. 131).

[79] Steven D Hales. 'Ockham's Disposable Razor'. In: *The Role of Pragmatics in Contemporary Philosophy: Contributions of the Austrian Ludwig Wittgenstein Society* (1997), pp. 356–361 (cit. on p. 10).

[80] Terry Halpin. 'Object-role modeling (ORM/NIAM)'. In: *Handbook on architectures of information systems*. Springer, 2006, pp. 81–103 (cit. on p. 138).

[81] Harvard University, Academia Sinica and Peking University. *China Biographical Database (CBDB)*. Jan. 2023. URL: `https://projects.iq.harvard.edu/cbdb` (cit. on p. 131).

[82] Frank Herbert. *Dune*. Penguin, 1999 (cit. on p. 147).

[83] Joachim Herbst. 'A Machine Learning Approach to Workflow Management'. In: *Machine Learning: ECML 2000*. Ed. by Ramon López de Mántaras and Enric Plaza. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2000, pp. 183–194. ISBN: 978-3-540-45164-8. DOI: `10.1007/3-540-45164-1_19` (cit. on p. 5).

[84] J. Hillston. 'Process algebras for quantitative analysis'. In: *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*. ISSN: 1043-6871. June 2005, pp. 239–248. DOI: `10.1109/LICS.2005.35` (cit. on p. 86).

[85] Charles O Hucker. *A Dictionary of Official Titles in Imperial China.* Peking University Press, 2008 (cit. on p. 131).

[86] Gert Janssenswillen, Benoıt Depaire and Christel Faes. 'Enhancing Discovered Process Models using Bayesian Inference and MCMC'. In: *Proceedings of the 2020 BPI Workshop.* 2020, pp. 295–307 (cit. on pp. 6, 28, 42).

[87] Gert Janssenswillen, Niels Donders, Toon Jouck and Benoît Depaire. 'A comparative study of existing quality measures for process discovery'. en. In: *Information Systems* 71 (Nov. 2017), pp. 1–15. ISSN: 0306-4379. DOI: 10.1016/j.is.2017.06.002. URL: http://www.sciencedirect.com/science/article/pii/S030643791730145X (visited on 16/12/2020) (cit. on p. 105).

[88] Ian Jolliffe. 'Principal Component Analysis'. en. In: *International Encyclopedia of Statistical Science.* Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer, 2011, pp. 1094–1096. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_455. URL: https://doi.org/10.1007/978-3-642-04898-2_455 (visited on 05/07/2021) (cit. on p. 116).

[89] Anna Kalenkova, Artem Polyvyanyy and Marcello La Rosa. 'A Framework for Estimating Simplicity of Automatically Discovered Process Models Based on Structural and Behavioral Characteristics'. en. In: *Business Process Management.* Ed. by Dirk Fahland, Chiara Ghidini, Jörg Becker and Marlon Dumas. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 129–146. ISBN: 978-3-030-58666-9. DOI: 10.1007/978-3-030-58666-9_8 (cit. on pp. 8, 33, 43, 91, 120).

[90] Krzysztof Kluza, Grzegorz J. Nalepa and Janusz Lisiecki. 'Square Complexity Metrics for Business Process Models'. en. In: *Advances in Business ICT.* Ed. by Maria Mach-Król and Tomasz Pełech-Pilichowski. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2014, pp. 89–107. ISBN: 978-3-319-03677-9. DOI: 10.1007/978-3-319-03677-9_6. URL: https://doi.org/10.1007/978-3-319-03677-9_6 (visited on 01/09/2020) (cit. on p. 33).

[91] D.E. Knuth. *The Art of Computer Programming: Volume 3: Sorting and Searching.* Pearson Education, 1998. ISBN: 978-0-321-63578-5. URL: https://books.google.com.au/books?id=cYULBAAAQBAJ (cit. on p. 100).

[92] Christian Kohlschein. *An introduction to hidden Markov models.* Tech. rep. Aachen, Germany, 2006 (cit. on p. 5).

[93] Hyunsoo Lee, Hongsuk Park and Amamath Banerjee. 'Representation, simulation and control of manufacturing process with different forms of uncertainties'. In: *Winter Simulation Conference.* WSC '09. Austin, Texas: Winter Simulation Conference, Dec. 2009, pp. 2262–2271. ISBN: 978-1-4244-5771-7. (Visited on 03/03/2020) (cit. on p. 11).

[94] Sander J. J. Leemans. 'Leveraging frequencies in event data: a pledge for stochastic process mining'. In: *Workshop on Algorithms and Theories for the Analysis of Event Data.* 2022 (cit. on p. 1).

[95] Sander J. J. Leemans, Wil M. P. van der Aalst, Tobias Brockhoff and Artem Polyvyanyy. 'Stochastic process mining: Earth movers' stochastic conformance'. en. In: *Information Systems* 102 (Feb. 2021), p. 101724. ISSN: 0306-4379. DOI: `10.1016/j.is.2021.101724`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437921000041` (visited on 14/02/2021) (cit. on pp. 6, 7, 9, 12, 24, 28, 33, 34, 41, 43, 81, 82, 90, 106, 108, 109, 114, 132, 148, 156).

[96] Sander J. J. Leemans, Dirk Fahland and Wil M. P. van der Aalst. 'Discovering Block-Structured Process Models from Event Logs - A Constructive Approach'. en. In: *Application and Theory of Petri Nets and Concurrency*. Ed. by José-Manuel Colom and Jörg Desel. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 311–329. ISBN: 978-3-642-38697-8. DOI: `10.1007/978-3-642-38697-8_17` (cit. on pp. 6, 34, 42, 73, 167).

[97] Sander J. J. Leemans, Dirk Fahland and Wil M. P. van der Aalst. 'Scalable process discovery and conformance checking'. en. In: *Software & Systems Modeling* 17.2 (May 2018), pp. 599–631. ISSN: 1619-1374. DOI: `10.1007/s10270-016-0545-x`. URL: `https://doi.org/10.1007/s10270-016-0545-x` (visited on 26/07/2020) (cit. on pp. 6, 42, 47).

[98] Sander J. J. Leemans, Fabrizio Maria Maggi and Marco Montali. 'Reasoning on Labelled Petri Nets and Their Dynamics in a Stochastic Setting'. en. In: *Business Process Management*. Ed. by Claudio Di Ciccio, Remco Dijkman, Adela del Río Ortega and Stefanie Rinderle-Ma. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 324–342. ISBN: 978-3-031-16103-2. DOI: `10.1007/978-3-031-16103-2_22` (cit. on pp. 7, 12, 24, 25, 90, 101, 106).

[99] Sander J. J. Leemans, Lisa L. Mannel and Natalia Sidorova. 'Significant stochastic dependencies in process models'. en. In: *Information Systems* (May 2023), p. 102223. ISSN: 0306-4379. DOI: `10.1016/j.is.2023.102223`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437923000595` (visited on 12/05/2023) (cit. on pp. 6, 12, 37, 132).

[100] Sander J. J. Leemans and Artem Polyvyanyy. 'Stochastic-Aware Conformance Checking: An Entropy-Based Approach'. en. In: *Advanced Information Systems Engineering*. Ed. by Schahram Dustdar, Eric Yu, Camille Salinesi, Dominique Rieu and Vik Pant. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 217–233. ISBN: 978-3-030-49435-3. DOI: `10.1007/978-3-030-49435-3_14` (cit. on pp. 4, 9, 11, 12, 22, 27, 82, 114).

[101] Sander J. J. Leemans and Artem Polyvyanyy. 'Stochastic-aware precision and recall measures for conformance checking in process mining'. en. In: *Information Systems* 115 (May 2023), p. 102197. ISSN: 0306-4379. DOI: `10.1016/j.is.2023.102197`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437923000339` (visited on 16/03/2023) (cit. on pp. 7, 9, 12, 23, 28, 33, 34, 43, 55, 90, 102, 106, 110, 148, 150).

[102] Sander J. J. Leemans, Anja F Syring and Wil MP van der Aalst. 'Earth movers' stochastic conformance checking'. In: *International Conference on Business Process Management*. Springer, 2019, pp. 127–143 (cit. on pp. 7, 52, 93).

[103] Sander J.J. Leemans, Erik Poppe and Moe T. Wynn. 'Directly Follows-Based Process Mining: Exploration & a Case Study'. In: *2019 International Conference on Process Mining (ICPM).* June 2019, pp. 25–32. DOI: 10.1109/ICPM.2019.00015 (cit. on p. 6).

[104] Sander JJ Leemans, James M McGree, Artem Polyvyanyy and Arthur HM ter Hofstede. 'Statistical Tests and Association Measures for Business Processes'. In: *IEEE Transactions on Knowledge and Data Engineering* (2022). Publisher: IEEE (cit. on p. 8).

[105] Massimiliano de Leoni and Felix Mannhardt. *Road Traffic Fine Management Process.* Dataset. https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5. 2015 (cit. on pp. 16, 82).

[106] Veronica Liesaputra, Sira Yongchareon and Sivadon Chaisiri. 'Efficient Process Model Discovery Using Maximal Pattern Mining'. en. In: *Business Process Management.* Ed. by Hamid Reza Motahari-Nezhad, Jan Recker and Matthias Weidlich. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 441–456. ISBN: 978-3-319-23063-4. DOI: 10.1007/978-3-319-23063-4_29 (cit. on p. 42).

[107] Chuang LIN, Yang QU, Fengyuan REN and Dan C Marinescu. 'Performance equivalent analysis of workflow systems based on stochastic petri net models'. In: *International Conference on Engineering and Employment of Cooperative Information Systems.* Springer, 2002, pp. 64–79 (cit. on p. 10).

[108] Fabrizio M. Maggi, Marco Montali and Rafael Peñaloza. 'Temporal Logics Over Finite Traces with Uncertainty (Technical Report)'. In: *arXiv:1903.04940 [cs]* (Nov. 2019). arXiv: 1903.04940. URL: http://arxiv.org/abs/1903.04940 (visited on 18/06/2020) (cit. on p. 7).

[109] Fabrizio Maria Maggi, Marco Montali and Rafael Peñaloza. 'Probabilistic Conformance Checking Based on Declarative Process Models'. en. In: *Advanced Information Systems Engineering.* Ed. by Nicolas Herbaut and Marcello La Rosa. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2020, pp. 86–99. ISBN: 978-3-030-58135-0. DOI: 10.1007/978-3-030-58135-0_8 (cit. on p. 7).

[110] Lisa L. Mannel and Wil M. P. van der Aalst. 'Discovering Process Models with Long-Term Dependencies While Providing Guarantees and Handling Infrequent Behavior'. en. In: *Application and Theory of Petri Nets and Concurrency.* Ed. by Luca Bernardinello and Laure Petrucci. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, pp. 303–324. ISBN: 978-3-031-06653-5. DOI: 10.1007/978-3-031-06653-5_16 (cit. on p. 38).

[111] Felix Mannhardt. *Sepsis Cases - Event Log.* Dataset. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460. 2016 (cit. on pp. 16, 82).

[112] Felix Mannhardt, Sander J. J. Leemans, Christopher T. Schwanen and Massimiliano de Leoni. 'Modelling Data-Aware Stochastic Processes - Discovery and Conformance Checking'. In: *Application and Theory of Petri Nets and Concurrency.* Ed. by Luis Gomes and Robert Lorenz. Cham: Springer Nature Switzerland, 2023, pp. 77–98. ISBN: 978-3-031-33620-1 (cit. on pp. 6, 28, 37, 39, 132, 151).

[113] Tabib Ibne Mazhar, Asad Tariq, Sander JJ Leemans, Kanika Goel, Moe T Wynn and Andrew Staib. 'Stochastic-Aware Comparative Process Mining in Healthcare'. In: *Business Process Management*. Utrecht, The Netherlands, 2023 (cit. on pp. 8, 11, 126).

[114] Jan Mendling, Gustaf Neumann and Wil van der Aalst. 'Understanding the Occurrence of Errors in Process Models Based on Metrics'. en. In: *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*. Ed. by Robert Meersman and Zahir Tari. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 113–130. ISBN: 978-3-540-76848-7. DOI: `10.1007/978-3-540-76848-7_9` (cit. on pp. 9, 115).

[115] Jan Mendling, Hajo A. Reijers and Jorge Cardoso. 'What Makes Process Models Understandable?' en. In: *Business Process Management*. Ed. by Gustavo Alonso, Peter Dadam and Michael Rosemann. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 48–63. ISBN: 978-3-540-75183-0. DOI: `10.1007/978-3-540-75183-0_4` (cit. on pp. 31, 105, 109).

[116] Rana Mitter. *China' s Good War: How World War II Is Shaping a New Nationalism*. Harvard University Press, 2020. ISBN: 978-0-674-98426-4. DOI: `10.2307/j.ctv322v52h`. URL: `https://www.jstor.org/stable/j.ctv322v52h` (visited on 03/02/2023) (cit. on p. 129).

[117] Catarina Moreira, Emmanuel Haven, Sandro Sozzo and Andreas Wichert. 'Process mining with real world financial loan applications: Improving inference on incomplete event logs'. en. In: *PLOS ONE* 13.12 (Dec. 2018). Publisher: Public Library of Science, e0207806. ISSN: 1932-6203. DOI: `10.1371/journal.pone.0207806`. URL: `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0207806` (visited on 03/11/2020) (cit. on pp. 6, 12, 105).

[118] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994 (cit. on p. 31).

[119] John F Padgett. 'Hierarchy and ecological control in federal budgetary decision making'. In: *American Journal of Sociology* 87.1 (1981). Publisher: University of Chicago Press, pp. 75–129 (cit. on p. 11).

[120] Marco Pegoraro and Wil MP van der Aalst. 'Mining uncertain event data in process mining'. In: *2019 International Conference on Process Mining (ICPM)*. IEEE, 2019, pp. 89–96 (cit. on p. 10).

[121] Marco Pegoraro, Bianka Bakullari, Merih Seran Uysal and Wil M. P. van der Aalst. 'Probability Estimation of Uncertain Process Trace Realizations'. en. In: *Process Mining Workshops*. Ed. by Jorge Munoz-Gama and Xixi Lu. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2022, pp. 21–33. ISBN: 978-3-030-98581-3. DOI: `10.1007/978-3-030-98581-3_2` (cit. on pp. 10, 11).

[122] Marco Pegoraro, Merih Seran Uysal and Wil M. P. van der Aalst. 'Efficient Time and Space Representation of Uncertain Event Data'. en. In: *Algorithms* 13.11 (Nov. 2020). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 285. DOI: `10.3390/a13110285`. URL: `https://www.mdpi.com/1999-4893/13/11/285` (visited on 10/12/2020) (cit. on pp. 10, 11).

[123] Marco Pegoraro, Merih Seran Uysal and Wil MP van der Aalst. 'Discovering process models from uncertain event data'. In: *International Conference on Business Process Management.* Springer, 2019, pp. 238–249 (cit. on pp. 10, 11).

[124] Carl Adam Petri. *Communication with automata.* Tech. rep. RADC-TR-65-377. Translation of PhD Thesis. New York: Grifiss Air Force Base, 1966 (cit. on p. 86).

[125] Sadie Plant. *Zeros and ones: Digital women and the new technoculture.* London, 1997 (cit. on p. 1).

[126] Artem Polyvyanyy, Alistair Moffat and Luciano García-Bañuelos. 'An entropic relevance measure for stochastic conformance checking in process mining'. In: *2020 2nd International Conference on Process Mining (ICPM).* IEEE, 2020, pp. 97–104 (cit. on pp. 6, 82).

[127] Artem Polyvyanyy, Sergey Smirnov and Mathias Weske. 'Reducing the complexity of large EPCs'. In: *Modellierung betrieblicher Informationssyteme (MobIS): Geschäftsprozessmanagement mit EPK.* Vol. 141. Lecture Notes in Informatics. Saarbrücken, Germany: Gesellschaft für Informatik, Nov. 2008, pp. 195–207 (cit. on p. 11).

[128] Hajo Reijers. 'What Have the Romans ever Done for Us? The Ancient Antecedents of Business Process Management'. In: *Business Process Management.* Keynote. Rome, 2021 (cit. on p. 132).

[129] Andreas Rogge-Solti. 'Probabilistic Estimation of Unobserved Process Events'. Doctoral Thesis. Potsdam, Germany: Universität Potsdam, 2014 (cit. on p. 6).

[130] Andreas Rogge-Solti, Wil M. P. van der Aalst and Mathias Weske. 'Discovering Stochastic Petri Nets with Arbitrary Delay Distributions from Event Logs'. en. In: *Business Process Management Workshops.* Ed. by Niels Lohmann, Minseok Song and Petia Wohed. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2014, pp. 15–27. ISBN: 978-3-319-06257-0. DOI: `10.1007/978-3-319-06257-0_2` (cit. on pp. 6, 12, 22, 27, 28, 32–34, 37, 42, 82, 105, 107, 113, 132, 148, 150, 167).

[131] Andreas Rogge-Solti and Mathias Weske. 'Prediction of business process durations using non-Markovian stochastic Petri nets'. en. In: *Information Systems* 54 (Dec. 2015), pp. 1–14. ISSN: 0306-4379. DOI: `10.1016/j.is.2015.04.004`. URL: `http://www.sciencedirect.com/science/article/pii/S0306437915000642` (visited on 26/06/2020) (cit. on pp. 10, 22, 27).

[132] A. Rozinat and Wil M.P. van der Aalst. 'Conformance checking of processes based on monitoring real behavior'. en. In: *Information Systems* 33.1 (Mar. 2008), pp. 64–95. ISSN: 0306-4379. DOI: `10.1016/j.is.2007.07.001`. URL: `https://www.sciencedirect.com/science/article/pii/S030643790700049X` (visited on 19/02/2021) (cit. on p. 10).

[133] Roberto Segala and Nancy Lynch. 'Probabilistic simulations for probabilistic processes'. In: *Nordic Journal of Computing* 2.2 (1995). Publisher: Publishing Association Nordic Journal of Computing, pp. 250–273 (cit. on p. 44).

[134] Arik Senderovich, Sander J. J. Leemans, Shahar Harel, Avigdor Gal, Avishai Mandelbaum and Wil M. P. van der Aalst. 'Discovering Queues from Event Logs with Varying Levels of Information'. en. In: *Business Process Management Workshops*. Ed. by Manfred Reichert and Hajo A. Reijers. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2016, pp. 154–166. ISBN: 978-3-319-42887-1. DOI: `10.1007/978-3-319-42887-1_13` (cit. on pp. 6, 12).

[135] Arik Senderovich, Andreas Rogge-Solti, Avigdor Gal, Jan Mendling, Avishai Mandelbaum, Sarah Kadish and Craig A Bunnell. 'Data-driven performance analysis of scheduled processes'. In: *International Conference on Business Process Management*. Springer, 2016, pp. 35–52 (cit. on pp. 6, 12).

[136] Arik Senderovich, Alexander Shleyfman, Matthias Weidlich, Avigdor Gal and Avishai Mandelbaum. 'To aggregate or to eliminate? Optimal model simplification for improved process performance prediction'. In: *Information Systems* 78 (Nov. 2018), pp. 96–111. ISSN: 0306-4379. DOI: `10.1016/j.is.2018.04.003`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437916306329` (visited on 13/01/2024) (cit. on pp. 10, 42).

[137] Nuno Silva, Marcelo Silva, Carlos Mendes and Miguel Mira da Silva. 'A Public Organization Career Progression Analysis Using Process Mining'. In: *Atas da Conferência da Associação Portuguesa de Sistemas de Informação*. Vol. 15. Issue: 15. 2017, pp. 270–290 (cit. on p. 132).

[138] Ricardo Silva, Jiji Zhang and James G. Shanahan. 'Probabilistic workflow mining'. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. KDD '05. Chicago, Illinois, USA: Association for Computing Machinery, Aug. 2005, pp. 275–284. ISBN: 978-1-59593-135-1. DOI: `10.1145/1081870.1081903`. URL: `http://doi.org/10.1145/1081870.1081903` (visited on 19/02/2020) (cit. on pp. 5, 12).

[139] Elliott Sober. *Ockham' s Razors: A User' s Manual*. Cambridge University Press, 2015 (cit. on pp. 9, 105, 120).

[140] J Michael Spivey and JR Abrial. *The Z notation*. Vol. 29. Prentice Hall Hemel Hempstead, 1992 (cit. on p. 19).

[141] Ward Steeman. *BPI Challenge 2013*. Dataset. https://doi.org/10.4121/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07. 2014 (cit. on pp. 16, 82).

[142] N. Stephenson. *Termination Shock*. HarperCollins, 2021. ISBN: 978-1-4607-1296-2. URL: `https://books.google.com.au/books?id=p38sEAAAQBAJ` (cit. on p. 27).

[143] S. Suriadi, R. Andrews, A. H. M. ter Hofstede and M. T. Wynn. 'Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs'. en. In: *Information Systems* 64 (Mar. 2017), pp. 132–150. ISSN: 0306-4379. DOI: `10.1016/j.is.2016.07.011`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437915301344` (visited on 31/01/2023) (cit. on p. 141).

[144] Niek Tax, Xixi Lu, Natalia Sidorova, Dirk Fahland and Wil M. P. van der Aalst. 'The imprecisions of precision measures in process mining'. In: *Inf. Process. Lett.* 135 (2018), pp. 1–8. DOI: `10.1016/j.ipl.2018.01.013`. URL: `https://doi.org/10.1016/j.ipl.2018.01.013` (cit. on pp. 9, 61).

[145] Niek Tax, Natalia Sidorova, Reinder Haakma and Wil MP van der Aalst. 'Mining local process models'. In: *Journal of Innovation in Digital Ecosystems* 3.2 (2016). Publisher: Elsevier, pp. 183–196 (cit. on p. 101).

[146] Yann Thierry-Mieg. 'Structural Reductions Revisited'. en. In: *Application and Theory of Petri Nets and Concurrency.* Ed. by Ryszard Janicki, Natalia Sidorova and Thomas Chatain. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 303–323. ISBN: 978-3-030-51831-8. DOI: `10.1007/978-3-030-51831-8_15` (cit. on p. 42).

[147] Franck Thollard, Pierre Dupont and Colin De La Higuera. 'Probabilistic DFA inference using Kullback-Leibler divergence and minimality'. In: *In Seventeenth International Conference on Machine Learning.* Morgan Kaufmann, June 2000, pp. 975–982 (cit. on pp. 5, 12, 42).

[148] Loukas C. Tsironis, Dimitris S. Sfiris and Basil K. Papadopoulos. 'Fuzzy Performance Evaluation of Workflow Stochastic Petri Nets by Means of Block Reduction'. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 40.2 (Mar. 2010), pp. 352–362. ISSN: 1558-2426. DOI: `10.1109/TSMCA.2009.2035303` (cit. on pp. 10, 126).

[149] Vijay Vaishnavi, William Kuechler and Stacie Petter. *Design science research in information systems.* Tech. rep. (created in 2004 and updated until 2015 by Vaishnavi, V. and Kuechler, W.); last updated (by Vaishnavi, V. and Petter, S.), June 30, 2019. Association For Information Systems, June 2019. URL: `http://www.desrist.org/design-research-in-information-systems/` (cit. on p. 15).

[150] Sicco Verwer, Rémi Eyraud and Colin De La Higuera. 'PAutomaC: a probabilistic automata and hidden Markov models learning competition'. In: *Machine learning* 96.1-2 (2014). Publisher: Springer, pp. 129–154 (cit. on pp. 5, 42).

[151] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta and R.C. Carrasco. 'Probabilistic finite-state machines - part I'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.7 (July 2005), pp. 1013–1025. ISSN: 1939-3539. DOI: `10.1109/TPAMI.2005.147` (cit. on pp. 23, 144).

[152] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta and R.C. Carrasco. 'Probabilistic finite-state machines - part II'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.7 (July 2005). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1026–1039. ISSN: 1939-3539. DOI: `10.1109/TPAMI.2005.148` (cit. on pp. 5, 23, 28, 42, 43, 55, 63, 90).

[153] Ulrike Von Luxburg and Bernhard Schölkopf. 'Statistical Learning Theory: Models, Concepts, and Results'. en. In: *Handbook of the History of Logic*. Ed. by Dov M. Gabbay, Stephan Hartmann and John Woods. Vol. 10. Inductive Logic. North-Holland, Jan. 2011, pp. 651–706. DOI: `10.1016/B978-0-444-52936-7.50016-1`. URL: `https://www.sciencedirect.com/science/article/pii/B9780444529367500161` (visited on 15/02/2021) (cit. on pp. 10, 105).

[154] Jianmin Wang, Shaoxu Song, Xiaochen Zhu and Xuemin Lin. 'Efficient recovery of missing events'. In: *Proceedings of the VLDB Endowment* 6.10 (2013). Publisher: VLDB Endowment, pp. 841–852 (cit. on pp. 14, 25).

[155] Xiaoli Wang, Guangju Chen, Qiang Zhao and Zhongping Guo. 'Reduction of Stochastic Petri Nets for Reliability Analysis'. In: *2007 8th International Conference on Electronic Measurement and Instruments*. ISSN: null. Aug. 2007, pp. 1–222–1–226. DOI: `10.1109/ICEMI.2007.4350428` (cit. on p. 42).

[156] Yifang Wang, Hongye Liang, Xinhuan Shu, Jiachen Wang, Ke Xu, Zikun Deng, Cameron Dougall Campbell, Bijia Chen, Yingcai Wu and Huamin Qu. 'Interactive Visual Exploration of Longitudinal Historical Career Mobility Data'. In: *IEEE Transactions on Visualization and Computer Graphics* (2021). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 1–1. ISSN: 1941-0506. DOI: `10.1109/TVCG.2021.3067200` (cit. on p. 132).

[157] Akio Watanabe, Yousuke Takahashi, Hiroki Ikeuchi and Kotaro Matsuda. 'Grammar-Based Process Model Representation for Probabilistic Conformance Checking'. In: *2022 4th International Conference on Process Mining (ICPM)*. IEEE, 2022, pp. 88–95 (cit. on pp. 7, 12, 90, 101).

[158] Jan Martijn E. M. van der Werf, Artem Polyvyanyy, Bart R. van Wensveen, Matthieu Brinkhuis and Hajo A. Reijers. 'All that glitters is not gold: Four maturity stages of process discovery algorithms'. en. In: *Information Systems* 114 (Mar. 2023), p. 102155. ISSN: 0306-4379. DOI: `10.1016/j.is.2022.102155`. URL: `https://www.sciencedirect.com/science/article/pii/S0306437922001338` (visited on 28/07/2023) (cit. on pp. 14, 148).

[159] Jing Yang, Chun Ouyang, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede and Yang Yu. 'OrdinoR: A framework for discovering, evaluating, and analyzing organizational models using event logs'. en. In: *Decision Support Systems* 158 (July 2022), p. 113771. ISSN: 0167-9236. DOI: `10.1016/j.dss.2022.113771`. URL: `https://www.sciencedirect.com/science/article/pii/S0167923622000422` (visited on 12/05/2023) (cit. on p. 132).

[160] MengChu Zhou and Kurapati Venkatesh. *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. en. Google-Books-ID: KQlXHGAQtCIC. World Scientific, 1999. ISBN: 978-981-02-3029-6 (cit. on pp. 11, 126).