AN APPROACH FOR HANDLING CONCEPT DRIFT AND MODEL SIMPLIFICATION IN LOG-BASED PROCESS ANALYSIS

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

MANOJ KUMAR MUTTYAL VASANTHA KUMAR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE - 575025

NOVEMBER 2017

Dedicated to my beloved parents and teachers.

DECLARATION

by the Ph.D. Research Scholar

I hereby **declare** that the Research Thesis entitled **An Approach for Handling Concept Drift and Model Simplification in Log-Based Process Analysis** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Computer Science and Engineering** is a **bonafide report of the research work carried out by me**. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

(**CS12F06, Manoj Kumar M V**)
(Register Number, Name & Signature of Research Scholar)
Department of Computer Science and Engineering

Place: NITK, Surathkal.
Date: November 13, 2017

CERTIFICATE

This is to *certify* that the Research Thesis entitled **An Approach for Handling Concept Drift and Model Simplification in Log-Based Process Analysis** submitted by **Manoj Kumar Muttyal Vasanth Kumar**, (Register Number: **CS12F06**) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.

Dr. Annappa B
Research Guide
(Name and Signature with Date and Seal)

Chairman - DRPC
(Signature with Date and Seal)

Acknowledgments

In this incredible four years journey of my Ph.D., I had the fortune to meet and work with many outstanding people, without whom I never could have done it. Their encouragements have allowed me to push the envelope beyond what I originally thought to be feasible. For this reason, I would like to mention the individuals to whom I owe my genuine admiration and thankfulness.

I wouldn't know what was research and its depth if I was not given an opportunity to do it. A great share of credit is due to my research supervisor, **Dr. Annappa** who gave me an opportunity to work in the significant field. There are absolutely no words how much I am grateful to my guide who supported me throughout my journey of ups and downs. His constant support, guidance, supervision accorded me to focus on my research work. His unrelenting passion for quality and soundness in research, and his incredible professionalism, work ethic, and drive, have been a defining inspiration for my endeavors. Thank you sir for always leading me the right path.

I am enormously thankful to the Research Progress Assessment Committee members **Dr. Shashidhar G Koolagudi** and **Dr. Harsha Vardhan** for their insightful comments, critical questions, and valuable ideas. Their continuous interest about my research progress and their sharp and quick feedback in all matters has greatly helped me in achieving research related objectives.

I should thank **Prof. K Chandrasekaran** for inspiring me in many ways during research tenure. His unparalleled commitment towards research and teaching had influenced me in the way that I cannot describe in words. I am thankful to **Dr. Mohith Tahilani** for providing convincing solutions for research and academics related matters. I should be grateful **Dr. Chandavardkar** and **Mrs. Soumya Hegde** for their cheering words. I humbly thank **Dr. Santhi Thilagam**, HOD and **Dr. Alwyn Roshan Pais**, Chairman(DRPC) and **Dr. Manu Basavaraju**, Secretary (DRPC) for helping me in research related aspects. I should be thankful for the support and advices received from **Dr. Jeny Rajan**, **Mrs. Vani M** and **Dr. Basavaraj Talwar**.

My journey during Ph.D. wouldn't have been exciting if I don't have this particular friend **Mr. Likewin Thomas**. He shares the 99.9% of my memories at NITK. I just cannot thank him in a single paragraph. He helped me in many aspects right from the research to life. We have shared many quiet and cheerful moments. With him I have traveled, cooked, written papers, did night outs, worked for conferences and workshops, and what not..!(this listing is virtually never-ending). It has been a genuine pleasure working with you!

I should be thankful to **Mrs. Priyanka Madukar** for thoroughly reading through my text and providing me with detailed comments and critical remarks. Feedbacks and bits of advice given by her helped me in improving the quality of my research related documents to many folds.

My cubicle/research lab was more exciting and fun with many colleague/friends around me. **Praveen** my late night lab mate, **Vishnu** who never fails to bring a smile on my face and such a great person to work with, **Keerthi**, a good fun loving friend who makes the place come alive. **Sachin**, **Sumith**, **Bane Rahman**, **Vishal**, **Girish**, **Bhimappa**, **Khyamling**, **Manjunath Mulimani**, **Marimuttu**, **Raghavan**, and **Fathima** made the place so lively and educative. They managed to make me smile whenever I was down. They held my hand through bad times. It is my pleasure to share the lab with you kind people, you all contributed, big or small, to this work.

I became friends with **Puneeth** and **Prashanth** during my masters. It has been more than half a decade we are the best friends. They always took my side during every moment of joy and sorrow. I am grateful to them for many aspects . Thank you both for the excellent support, fun-filled moments and every other thing. **Manoj Revankar** is my decade old friend, and knows all my secrets and shares almost every memorable moments of PU, undergrad and postgraduate Thank you, Manoj for always bringing a smile on my face.

I would humbly thank the non-teaching staff of my department. **Mr. Dinesh Kamath**, **Mrs. Yashawanthi** and **Mr. Vairavanathan** were supportive in ensuring that research-related seminars go well and uninterrupted. I should be thankful to

Ms. Vanitha, **Mrs. Seema Shivaram**, and **Mrs. Mohini** for acknowledging me through the academic work and helping me the all the time. **Ms. Krithi** and **Mr. Ravi** were wonderful in helping me for several times. I can't forget **Mr. Dayanand** for his help regarding academic and office related matters.

Sneha is my best friend for almost a more than five years now. During not-so-great times of my life, she was right there and provided excellent support. She has always been an integral part of my every success. I dont have words to express my gratitude for the enormous amount of time, support and advice she gave me whenever I needed it. She always motivated me to strive for the best. It is tough to imagine a better friend than her. Thank you, Sneha.

I want to thank my parents **Mr. Vasantha Kumara** and **Mrs. Meenakshi** for being the main pillar of this journey and for constant and unconditional love towards me. They understood and encouraged me in all my decisions. Without their support, this thesis would not have been possible. Even though I was away from them for these four years, they managed to pray for me and stay calm throughout my stay in NITK. Without their constant encouragement, this journey would be left incomplete. With this, I would like to thank all of my family members for all the support and encouragement I have got always.

In the end, all that remains is to thank you, dear reader. If you have found at least a small part of this thesis useful or interesting, you have made all my work worthwhile

Finally my inmost gratitude towards almighty for helping me get through this!

Place: Surathkal **Manoj Kumar M V**

Date: November 13, 2017

Abstract

Nowadays process aware information systems supporting operational processes are in the mainstream. Examples are work-flow management systems, enterprise application integration systems, enterprise resource planning systems, web services, etc. These systems are recording detailed information about the history of processes execution in the form of events. Events may range from the withdrawal of cash from an ATM, a doctor adjusting an X-ray machine, a citizen applying for a driver license, the submission of a tax declaration, and the receipt of an e-ticket number by a traveler. The challenge is to extract knowledge out of event data for improving the original processes in a meaningful way.

A young research discipline named process mining offers a spectrum of techniques for analyzing event data generated in information systems. These methods can be seen as the amalgamation of computational intelligence and data mining on the one hand, and process modeling and analysis on the other hand. It offers a variety of techniques for discovering, monitoring and improving processes in numerous application domains. Through this research work, most significant issues present in process mining are identified and the practical solutions are proposed. The techniques addressing these issues are classified into following three broad categories.

First category of techniques concentrate on addressing non-stationary learning problem known as *concept drift*. Concept drift is a phenomenon when process changes dynamically during the period of execution and/or analysis. Due to this, state-of-the-art process mining techniques generate inconsistent and obsolete analysis results. Therefore, it is required to design and implement the methods which can efficiently address concept drift. We have proposed a Multiple Trace Alignment method for detecting and localizing concept drift in control-flow perspective of the operational process. The proposed method has been tested on real-life event log and compared with the existing methods for handling concept drift.

Second category of techniques concentrate on developing a notation named trace logo for visualizing control-flow perspective. Traditional control-flow discovery algorithms in process mining can generate process model consisting of activities and transitions and ignore all other information. But, the trace logo overcomes the drawbacks of traditional approaches, and it is capable of visualizing activities, transitions, the consensus of the traces, order of prevalence between activity, relative occurrences of every activity, information scores, set of conserved and shared activities/sequences in a single compact graphic.

Third category of techniques are centered on path discovery and complexity reduction in structured and unstructured processes. If the process is Lasagna (structured), feature set capturing the control-flow properties are extracted and used. If the process is of Spaghetti (unstructured), it is reduced to Lasagna and features capturing the control-flow properties are obtained. Feature sets are systematically analyzed to find the details like frequent, infrequent, possible, and impossible paths of executions in the process.

All the proposed methods in this thesis are evaluated on real-life event log taken from standard repository and results are presented in subsequent chapters. Through this research work, a most sincere and prompt effort has been made to leverage the existing process mining practices by addressing the diverse categories of most significant issues.

Table of Contents

List of Figures

List of Tables

Abbreviations and Nomenclature

Abbreviations

AGNEs	Artificially Generated Negative Events
AHC	Agglomerative Hirarchical Clustering
AI	Artificial Intelligence
AUC	Area Under Curve
BPM	Business Process Management
BPMN	Business Process Modeling Notation
Case Id	Case Identifier
CPN	Colored Petri Net
CPN XES	Colored Petri Net eXtensible Event Stream
CRM	Customer Relationship Management
EMiT	Enhanced Mining Tool
EPC	Event-Driven Process Chain
ERP	Enterprise Resource Planning
Event Id	Event Identifier
IEEE CIS	Institute of Electrical and Electronics Engineers Computational Intelligence Society
ILP Miner	Integer Linear Programming Miner

IM and IM_i	Inductive Miner and Inductive Miner - infrequent
IM_i	Inductive Miner - infrequent
MiMo	Mining Module
MiSoN	Mining Social Networks
MTA	Multiple Trace Alignment
MXML	Mining eXtensible Markup Language
PAIS	Process Aware Information Systems
PDM	Produce Data Management
ProM	Process Mining Framework
ROC	Receiver Operating Characteristic
SA-MXML	Symantically Annotated Mining eXtensible Markup Language, Artificially Generated Negative Events
WFM	Workflow Management
XES	eXtensible Event Streams
XML	Extensible Markup Language
XOR	Exclusive OR
YAWL	Yet Another Workflow Language

Nomenclature

$\#_{activity}(e)$	Event log classifier for activity attribute
$\#_{resource}(e)$	Event log classifier for the resource attribute
$\#_{time}(e)$	Event log classifier for the timestamp attribute
$\#_{trans}(e)$	Event log classifier for the transaction attribute

α	Significant probability
$\bar{\mathbb{T}}$	Mapped set of traces
$\bar{T}_i$	Transformed i^{th} trace
β	Total number of networks
$\cup$ and $\cap$	Union and intersection
$\emptyset$	Empty set
$\exists$	Exists
$\in$	Belongs
$\mathbb{B}$	Bag
$\mathbb{N}$	Set of all natural numbers
$\mathbb{N}^+$	Set of all positive natural numbers
$\mathbb{T}$	Set of traces
$\mathcal{A}$ and $\mathcal{A}^*$	Activity set and Activity sequence
$\mathcal{A}_{cd_i}$	Set of possible traces at sub-process level
$\mathcal{H}_0$ and $\mathcal{H}_1$	Null and alternative hypothesis
$\mathcal{L}$	Event log
$\mathcal{P}$	Power set
$\mathcal{P}_{cd_i}$	Set of possible traces at process level
$\mathcal{T}^n$	Trace of lenght n
$\setminus$	Set difference
σ	Trace
$\subseteq$ and $\subset$	Subset and Proper Subset

$\sum$ and $\sum^+$	Set of activities and Non emptry finite traces over $\sum$
$\wedge$ and $\vee$	And and Or
aRb	Relationship
cd_i	i^{th} concept drift
$Count_{always}$	Number of activities always follow a given activity
$Count_{never}$	Number of activities never follow a given activity
$Count_{sometimes}$	Number of activities sometimes follow a given activity
$Dom(R)$	Domain of relation R
e_n	Small-sample correction
$f : A \to B$	Function from set A to B
$f_{a,i}$	Relative frequency of activity $\mathtt{a}$ at position i
f_{rc}	Relationship count function
f_{re}	Relation entropy function
f_{wc}	Window count function
H_i	Uncertainity
I_l	Indel function
l_{max}	Maximum length of traces
l_{sum}	Sum of lengths of all traces
M_0, M_1,..	Total number of networks
n	Number of traces in the process
R_i	Information score
$Rng(R)$	Range of relation R

Chapter 1

Introduction

In this modern age, data is the new oil. Every second a new data is being created, the data volumes are exploding, more data has been created in the past two years than in the entire history of the human race (Petter, 2015). The rapid development of digital universe has brought us into the era of "big data". For example, E-bay, the online auction and web shopping company, processes around 50 petabytes of information daily. Facebook, a social networking platform, processes more than 500 terabytes of data daily. By the year 2020, the prediction is that each person would generate about 7 megabytes of new information every second (Bernard, 2017). A cloudburst of digital data are produced daily and will continue to increase exponentially in coming days.

At the moment less than 0.5% of all data is ever analyzed and used (Antonio, 2016). The immediate need is to extract and use the knowledge and information out of the recorded data. The knowledge gained by analyzing the raw data could play an invaluable role in a countless number of realms. One such aspect where data plays a key role (in understanding, implementing and improving) is business process/es in and across organizations.

The increased level of competition compelling individual organizations to perform in the best way possible. In the case of typical Fortune 1000 business organizations, just a 10% increase in data accessibility will results in $65 million additional net income (Dennis, 2014). A 65% of the senior executives are in agreement with the fact that management decisions are increasingly based on hard analytic information. This justifies the potential behind rightly utilizing the data. Availability of data and the capability of analyzing it could fetch fortunes in any realm. Depending on the context of application and problem at the hand, a variety of analysis techniques would give interesting and useful information.

Majority of organizations such as hospitals, industries, colleges, etc. are increasingly moving towards digitizing (or automating) their day-to-day processes with the help of information systems. These systems typically support logging capabilities that register what has been executed in the organization in the form of event log. Event logs usually contain,

- Data about cases (i.e. process instances) that have been executed in the organization.
- Times at which the tasks were executed.
- Persons or systems that performed these tasks.
- Additional process specific information.

Event logs offer a wealth of information for today's organizations, but it is rarely exploited to provide meaningful insights into business processes.

An interesting class of information systems that produce event logs are Process-Aware Information Systems (PAISs) (Dumas et al., 2005; Reichert and Weber, 2012) (we will use the generic term Process-Aware Information System to refer to systems that manage and execute such workflows) . Some of the widely used PAISs are classical workflow management systems (e.g. Staffware) (Yen, 2003), Enterprise Resource Planning systems (ERP) (e.g. Systems, Applications, Products) (Arik Ragowsky, 2002), case handling systems (e.g. FLOWer) (Van Der Aalst et al., 2005), Product Data Management systems (PDM) (e.g. Windchill) (Liu and Xu, 2001), Customer Relationship Management systems (CRM) (e.g. Microsoft Dynamics CRM) (Symons, 2014), middleware (e.g., IBM WebSphere) (De La Cruz et al., 2007), hospital information systems (e.g., Chipsoft) etc.

Nowadays most of the embedded systems (an embedded system is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls) are also capable of generating the event logs. An example is "CUSTOMerCARE Remote Services Network" of Philips Health care, where events occurring within an X-ray machine (e.g. moving the table, setting the deflector, etc.) are recorded and analyzed. These examples show that systems record (parts of) the actual behavior, and thus implicitly or explicitly create event logs.

Event log differs significantly from the standard tabular data (rows as observations and columns as attributes) in various aspects and simultaneously demands different analysis approaches. Experiences from data mining and various other research disciplines fail if applied for analyzing event logs due to following characteristics.

- Business operational processes potentially exhibiting concurrency are incomparable to simple data mining structures for example, decision trees and association rules.

- Data mining mainly focuses on the case perspective, i.e., cases with attribute values are analyzed without constructing some process model.

The goal is to develop the methods that are end-to-end and process centric, which facilitate us in analyzing event logs. A young data analysis horizon named ***process mining*** offers a spectrum of methods for analyzing event logs. It acts as a bridge between data mining and operational process modeling and analysis. The major concentration is on extracting the value and insights from these event data recorded in the form of the event log.

1.1 Process Mining

By using historical facts, as recorded by the information system, process mining provides detailed insights into the process execution. Process mining bridges the gap between the process-oriented nature of Business Process Management (BPM) and the data-oriented nature of machine learning/data mining. It is a research discipline that discovers, monitors, and improves real processes (not the assumed processes) by extracting knowledge from event logs readily available from today's systems (Van Der Aalst, 2011). The methods facilitated by process mining connects the modeled and observed behavior of the process.

The advantage of using process mining is two fold.

- It offers information on how processes are carried out in the real-world settings.

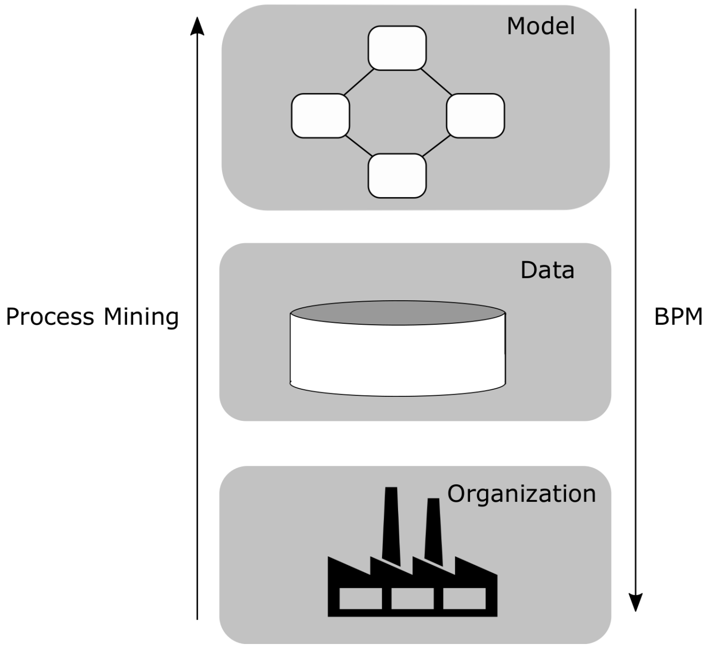


Figure 1.1: Relationship between process mining and BPM.

- Possibility to compare the actual behavior with the intended one. The expected behavior is usually defined either in a formal way (by defining a formal process model) or in an informal way. By comparing the models of expected behavior with the discovered process models, deviations from the intended behavior can be found, and this information can be used to improve the processes.

Approach followed by process mining is a reverse version of BPM (shown in the figure 1.1). BPM usually starts with high-level process design, followed by configuration and finally implementation. As in the case of process mining, observed behavior recorded in information systems are utilized to discover the real processes.

1.1.1 Basic types of techniques

The basic idea of process mining shown in the figure 1.2. Where the information systems supporting and controlling various real world processes record event log. With the availability of event log, different process mining analysis techniques can be employed to model and analyze the observed behavior.

There are three basic types of process mining techniques: discovery, conformance, and enhancement. For the complete framework of the ten different categories of techniques in process mining, readers are referred to read the chapter "Analyzing Lasagna Processes" in process mining book (Van Der Aalst, 2011).

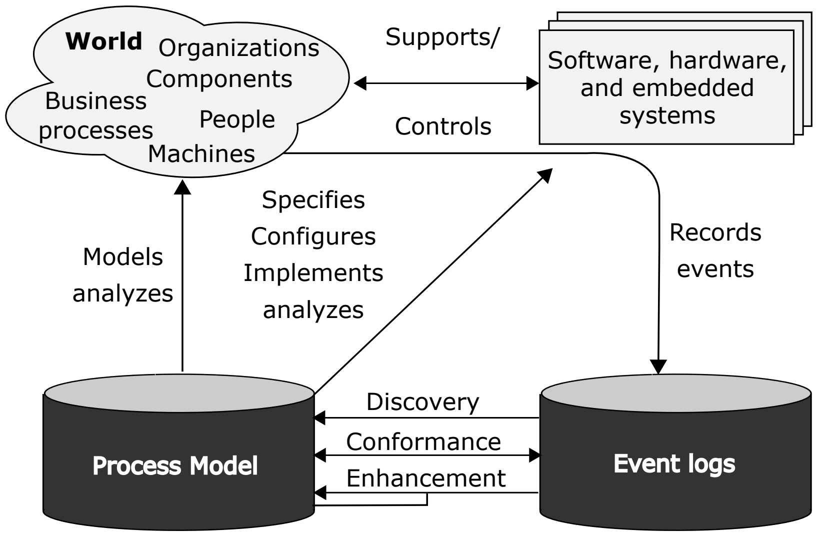


Figure 1.2: Basic types of process mining techniques (taken from Van Der Aalst (2011)).

Process ***discovery*** technique takes an event log as input and produces a model that best describes the behavior observed in the event log. Process discovery methods are mostly useful to provide insights into what occurs in reality. Discovery techniques produce control-flow, data, organizational, time, and case models. A plethora of methods exists in process mining to elicit process model in various modeling notations. For example, Fig. 2.2 shows control-flow model of hospital admission process discovered using α algorithm (Van Der Aalst et al., 2004).

Conformance checking techniques take a process model and an event log of the same process as input. Conformance measures and quantifies, how closely a given process model conforms to reality (as recorded in the event log). It evaluates the quality of the discovery algorithm and constructed model. Simplicity, precision, fitness, and generalization are the dimensions used for measuring the conformance.

The idea of ***enhancement*** is to improve the discovered process model to make it more informative. Various types of enhancement that one can perform, such as repairing a process model to better reflect the reality. Another type of enhancement is the extension of process models with information extracted from event logs. For example, a control-flow model can be enhanced and made more readable by overlaying the additional information such as time-stamps, bottlenecks, service levels, throughput times, frequencies, resources, decision rules quality metrics, etc.

1.2 Opportunities and Solutions

With the advent of process mining, analyzing operational process has become much meaningful and easier than before. Process mining enabled the analysis of operational process from the various perspectives such as control flow, data, organization, case, time, etc. Each of these perspectives delivers a particular piece of information unique to a process. In this thesis, we explore the control-flow perspective. We propose the methods and techniques which leverage the currently available practices related to control-flow and assist them in obtaining more robust solutions. The methods and techniques presented in this thesis are based on the following three categories of techniques.

1. The very first problem addressed in this thesis is to handle the situation of non-stationary learning problem named concept drift. Majority of process mining algorithms are stationary in nature. Due to concept drift, these algorithms generate obsolete analysis results. This thesis proposes a method for handling concept drift. Which can be seen as the missing component in the currently existing process mining algorithms. The proposed method can act as a necessary and off-the-shelf component in the upcoming algorithms of process mining.

2. In process mining, historically control flow perspective is depicted using connections between activities of the process (For example, Petri-net and Causal net process models). But there is no method for representing the consensus of the traces, the order of prevalence of the activities, information content, relative significance and conserved and shared activities/sequences of the process. We have developed a graphical notation named *trace logo* for addressing all the previously mentioned issues.

3. Capturing the operational process behavior and predicting the most frequent execution patterns in information systems has always been an matter of interest in process mining. Through *Position Specific Scoring Matrix* method we are able to capture the process state for detecting and predicting the most significant part of the control-flow.

1.3 Motivation

Following are most important facts and observations compelled us to explore our curiosity in this direction.

- The operational processes are the intellectual property of any organization. The method in which a particular task is carried out differs from an organization to the other. Business processes have to be adjusted to meet the demands, unforeseen events, peer competition and fierce modifications in the agreed service levels. For keeping an organization in the abreast of competition curve, change in the operational process becomes inevitable. Variations in the process are directly reflected in the event log. Currently, available process mining techniques stumble to produce stable analysis while analyzing event log of changed processes.

- ProM (Van Der Aalst et al., 2007a; Veck, 2016; Rubin et al., 2007) is the open source process mining framework offers more than 1200 algorithms in the form of plugins. But, the majority of these algorithms behave poorly and generate obsolete results when they are confronted with the process which has changed during the execution period.

- Currently available process mining algorithms assume that the operational process as a static entity and it does not change during the period of execution and analysis. But in reality, process tend to change in order to satisfy the intermittent needs, deadline, competition, and demands.

- Lack of taxonomy for classifying concept drift based on change types, patterns of change, perspectives, and mode of handling.

- Though there are several control-flow modeling methods are available, the information depicted by them are limited to activities and connections between activities. There is a need for the notation which could compactly represent control flow of a process with consensus, information content, relative significance, conserved activity or sequences details.

- There is a need for a method which could capture the control flow structure of a process. The obtained process structure could be used for detection and prediction of frequent activities and sequence of activities.

1.4 Research Contributions

Following contributions of this research work are available to the research community in the form of journal and conference publications.

- **Taxonomy for classifying different categories of concept drift:** It provides information for identifying classifying various categories of concept drift based on change type, change pattern, perspectives of change, mode of handling, sub problems and duration of change.

- **Framework for detecting control-flow concept drift:** Detecting any changes in the underlying process is the primary problem. For this, a complete event log is split into many small logs, and feature values embodying control-flow characteristics are extracted. Comparing features by employing statistical hypothesis reveals the control-flow drifts in sub-log level.

- **Localizing concept drift in control-flow perspective:** Localizing process change is the second step in handling concept drift. It aims at identifying the cases and traces affected by the change in the process. Trace specific feature set is constructed and analyzed to localize the changes.

- **Notation for illustrating control-flow more meaningfully:** A control-flow modeling notation named trace logo has been proposed. Unlike traditional approaches, trace logo is capable of displaying trace consensus, the relative significance of activities, conserved/shared activity/sequence, relative occurrence, and information score.

- **Identifying frequent paths of execution in Lasagna processes:** Finding the frequent paths of execution in Spaghetti process has been a matter of interest in the process mining research.

- **Simplifying Spaghetti structured process to Lasagna structure:** Spaghetti processes are unstructured, unreadable, and difficult to comprehend. The methods assisting in converting Spaghetti process to Lasagna process has been proposed.

- **Identifying and predicting possible and impossible paths of execution:** Methods and techniques have been demonstrated for identifying the possible and impossible paths of execution in Spaghetti structured processes. Identifying the set of possible and impossible traces help in optimizing the complex Spaghetti process in which they have numerous activities and paths.

1.5 Outline of the Thesis

This section briefly describes each chapter of this thesis, in order to give a brief overview about the structure.

- **Chapter 1**, the current chapter introduces the general domain of process mining and motivates the need for new mining techniques that are capable of handling concept drift, control-flow visualization, and complexity reduction.

- **Chapter 2** introduces the concepts and notations such as sets, lists, relations, functions, event logs, and method for producing artificial event logs that are needed for this thesis.

- **Chapter 3** describes the literature survey related to the research problem and objectives mentioned in this thesis.

- **Chapter 4** Introduces the method for detecting and locating sudden control flow the concept drift.

- **Chapter 5** proposes the method for constructing the control flow visualization notation named process logo.

- **Chapter 6** explains the method for simplifying the Spaghetti structured control flow to extract Lasagna structured control flow. It also explains about how to find the possible and impossible paths of execution in simplified Lasagna control-flow.

- **Chapter 7** proposes the methods and techniques for capturing the process execution information for identifying the frequent execution patterns.

- **Chapter 8** concludes the thesis and summarizes the contributions. Furthermore, the possible directions in which the proposed methods and techniques that can be improved are also briefly discussed.

Chapter 2

Preliminaries

In this chapter we discuss the necessary concepts and notations used in the upcoming contents of this thesis.

2.1 Sets, Lists, Relations and Functions

Definition 2.1. (Set notation)
A set is a infinite collection of elements. A finite set is denoted by listing its elements between braces. For example, a set S with elements a, b and c is represented as $\{a, b, c\}$. The empty set, i.e. the set with no elements, is denoted by ϕ.

By considering A and B as two sets, We define the following operations on sets.

- Number of elements in set A is denoted by $|A|$.
- $a \in A$ denotes that an element a is present in set A.
- $A \cap B$ denotes the intersection of sets A and B, is the set containing elements that are both in A and B: $A \cap B = \{x | x \in A \wedge x \in B\}$.
- $A \cup B$ denotes the union of two sets, is the set containing all elements that are in either A or B: $A \cup B = \{x | x \in A \vee x \in B\}$.
- $A \setminus B$ denotes difference between two sets. Is the set containing all elements of A that do not exist in B: $A \setminus B = \{x | x \in A \wedge x \notin B\}$.
- Set A becomes subset of B (denoted by $A \subseteq B$), if all elements of A are also in B. The set A is called proper subset (denoted by $A \subset B$) of B, if $A \subseteq B$, but not $A = B$. Power set of A denoted by $\mathcal{P}(A)$, it contains all possible subsets of A. If set A and B does not contain any common elements, then they are called as disjoint sets (denoted by $A \cap B = \phi$).

We use letter $\mathbb{N}$ for denoting set of all natural numbers (for example, $\mathbb{N} = \{0, 1, 2, 3...\}$) and $\mathbb{N}^+$ denotes the set of all positive natural numbers ($\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$).

Definition 2.2. (Cartesian product and ordered pairs)
The Cartesian product of two sets A and B is denoted as the set $|A \times B| = \{(a, b)|a \in A \wedge b \in B\}$. Where, first set in the Cartesian product (i.e., set A) is known as source set, and the second set (i.e., set B) is known as target set.

In a Cartesian product each element of a source set is related to every element of a target set. For example, Cartesian product of the sets $A = \{a, b\}$ and $B = \{1, 2\}$ is $\{(a, 1), (a, 2), (b, 1), (b, 2)\}$.

Definition 2.3. (Relation)
Let A and B be two sets. A relation is a subset of the Cartesian product, which relates some elements of A with some elements of B (denoted by aRb, where the set $R \subseteq A \times B$ is called a relation from A to B).

The domain of the relation is the set $Dom(R) = \{a \in A|\exists b \in B : aRb\}$. Its range is the set $Rng(R) = \{b \in B|\exists a \in A : aRb\}$. Relations that have the same source set and target set, we define the notions of reflexivity, symmetry, transitivity and antisymmetry. Let A be a set and let $R \subseteq A \times A$ be a relation, then,

- R is *reflexive* if aRa for all $a \in A$.
- R is *irreflexive* if $\sim (aRa)$ for all $a \in A$.
- If aRb implies bRa for all $a, b \in A$, the relation is *symmetric.*
- If aRb and bRc implies aRc for all a, b, c $\in A$, the relation is *transitive.*
- Relation R is *antisymmetric* if aRb and bRa imply $a = b$ for all $a, b \in A$.

Definition 2.4. (Functions)
A function is a relation that uniquely associates members of one set with members of another set (denoted by $f : A \to B$).

A function f from A to B is a relation such that for all $a \in A$ is individually linked with an object $f(a) \in B$. This implies that a function can be many-to-one or one-to-one relation. The set A of values at which a function is defined is called its domain, while the set $f(A) \subseteq B$ of values that the function can produce is called its range. Here, the set B is called the codomain of f.

Definition 2.5. (Multiset or Bag)
In a multiset (or bag), the members (elements) are allowed to appear more than once, whereas in a set the members can appear only once.

For example, a multiset consisting of two occurrences of a, two occurrences of b and single occurrence of c is denoted by $A = [a, a, b, b, c]$, same can also be represented as $[a^2, b^2, c]$. The superscript in the multiset representation signifies that how many times an element appeared (multiplicity of the particular element).

The empty bag, i.e. the bag for which all elements have multiplicity 0, is denoted by ϕ. If the multiplicity of an element is 0, we omit the element. Given a set A, multiset is defined as cardinal value function $M : A \rightarrow \mathbb{N}_0$ and $M(x) = 0$ if $x \notin A$. Set of all multisets over A is denoted by $\mathcal{B}(A) = A \rightarrow \mathbb{N}_0$.

Definition 2.6. (Sequences)
Let S be a set. A sequence σ over S of length $n \in \mathbb{N}$ is a function $\sigma : \{1, ..., n\} \rightarrow S$.

Given a sequence s and set A,

- Sequence s of length n are represented as $\langle s_1, s_2, s_3, .., s_n \rangle$.
- Length of sequence s is denoted by $|s|$.
- i^{th} symbol of s is represented by $s(i)$.
- A subsequence of s that starts at position i and ends at position j of s is represented by $s(i, j)$.
- A Head prefix of length i of s is represented by s^i it is equivalent to $s(1, i)$.
- A tail of a sequence s (suffix) that begins at position i is represented by $s(i, |s|)$.
- A new sequence p can be obtained by concatenating two sequences $s = \langle s_1, s_2, ..s_m \rangle$ and $t = \langle t_1, t_2, ..t_n \rangle$. After concatenation the resulting sequence p can be denoted by $s \Diamond t$, and it will be of length $m + n$.
- A^* represents the set of all finite sequences over A.
- Set of all sequences of length n over set A is denoted by A^n (is $\subseteq A^*$).

Definition 2.7. (Tuple)
Tuple (list) is a collection of ordered elements. Let A be a set and let $t = \langle a_1, a_2, ..., a_n \rangle \in A \times ... \times A$ be a tuple of n elements (generally called as n tuple). $t(i)$ refers to i^{th} element of tuple t. For example, let $\langle a, b \rangle \in A \times A$ be a tuple of 2 elements $t_1(\langle a, b \rangle) = a$ and $t_2(\langle a, b \rangle) = b$.

2.2 Event Log

The starting point of process mining is the observed behavior of process executions, stored in event logs. Since the event logs of information systems provide factual data about the underlying processes, they are a precious source of information. Primarily an event log consists of a bag of event traces (sequence of events that is recorded for a process instance is called a trace) and all traces are assumed to be from the single process. Same traces of events may occur multiple times indicating that different cases followed the same path in the process. Each line in the event log represents one event, and each column represents an attribute of this event. An event is associated with a case or process instance.

Table 2.1 shows an example of an event log of the hospital patient admission process. This event log will be used for discussion and illustration in this chapter. To better understand the structure of event log, a tree diagram of hospital patient admission event log is given in fig. 2.1. Using tree structure, we can describe the structure of event log as follows,

- A process involves numerous cases; a case comprises of events. Each event relates to the exactly single case, identified by case id.
- Events inside a case are well-ordered. Events can be explained by various attributes such as activity name, event id, timestamp, cost, data, resources, etc.

The typically seen information in the event log and their explanations are as follows.

- ***Case id*** is used for distinctly identifying a particular instance of the process. For example, ab12, ac13, etc. represent the case ids in the hospital admission event log given in table 2.1.
- ***Event id*** assigns a distinct identifier for every event related to a specific case. For example, event 2346 of case ab12 and event of 3347 of case ac13.

Case id	Event id	Properties				
		Timestamp	Activity	Resource	Cost	...
	2342	10-10-2012:01.20	`Appointment`	Anne	200	...
	2343	11-10-2012:07.24	`Admit patient`	Jhon	100	...
ab12	2344	12-10-2012:08.24	`Check history`	Thomas	400	...
	2345	13-10-2012:12.24	`Decide`	Clar	50	...
	2346	13-10-2012:13.25	`Begin treatment`	Ram	10	...
	3347	14-10-2012:01.29	`Appointment`	Anne	200	...
	3348	14-10-2012:01.34	`Out patient`	Wil	20	...
ac13	3349	14-11-2012:16.34	`Check history`	Thomas	300	...
	3350	15-11-2012:08.22	`Decide`	Clar	50	...
	3351	15-11-2012:09.17	`Discharge`	Ram	100	...
	4352	22-11-2012:07.47	`Appointment`	Anne	200	...
	4353	23-11-2012:12.31	`Admit patient`	John	100	...
	4354	25-11-2012:19.41	`Check history`	Thomas	50	...
	4355	25-11-2012:18.14	`Decide`	Clar	50	...
ad14	4356	25-11-2012:19.37	`Re-examine`	Steven	100	...
	4357	26-11-2012:01.00	`Out patient`	Wil	20	...
	4358	28-11-2012:02.00	`Check history`	Thomas	100	...
	4359	29-11-2012:13.37	`Decide`	Clar	400	...
	4360	30-11-2012:02.20	`Discharge`	Ram	100	...
	5361	30-12-2013:02.20	`Appointment`	Anne	200	...
	5362	11-01-2013:02.20	`Check history`	Thomas	300	...
ae15	5363	12-01-2013:03.20	`Admit patient`	Jhon	20	...
	5364	12-01-2013:15.20	`Decide`	Clar	50	...
	5365	13-01-2013:17.20	`Discharge`	Ram	30	...
	6366	17-01-2013:22.20	`Appointment`	Anne	200	...
	6367	17-01-2013:23.17	`Out patient`	Wil	50	...
	6368	19-01-2013:01.20	`Check history`	Thomas	100	...
	6369	19-01-2013:04.20	`Decide`	Clar	20	...
af16	6370	19-01-2013:15.20	`Re-initiate`	Steven	400	...
	6371	19-01-2013:22.20	`Admit patient`	Jhon	300	...
	6372	20-01-2013:05.20	`Check history`	Thomas	200	...
	6373	23-01-2013:07.20	`Decide`	Clar	300	...
	6374	24-01-2013:14.20	`Begin treatment`	Ram	200	...
...	...	...	...	...	...	...

Table 2.1: Event log of hospital admission process.

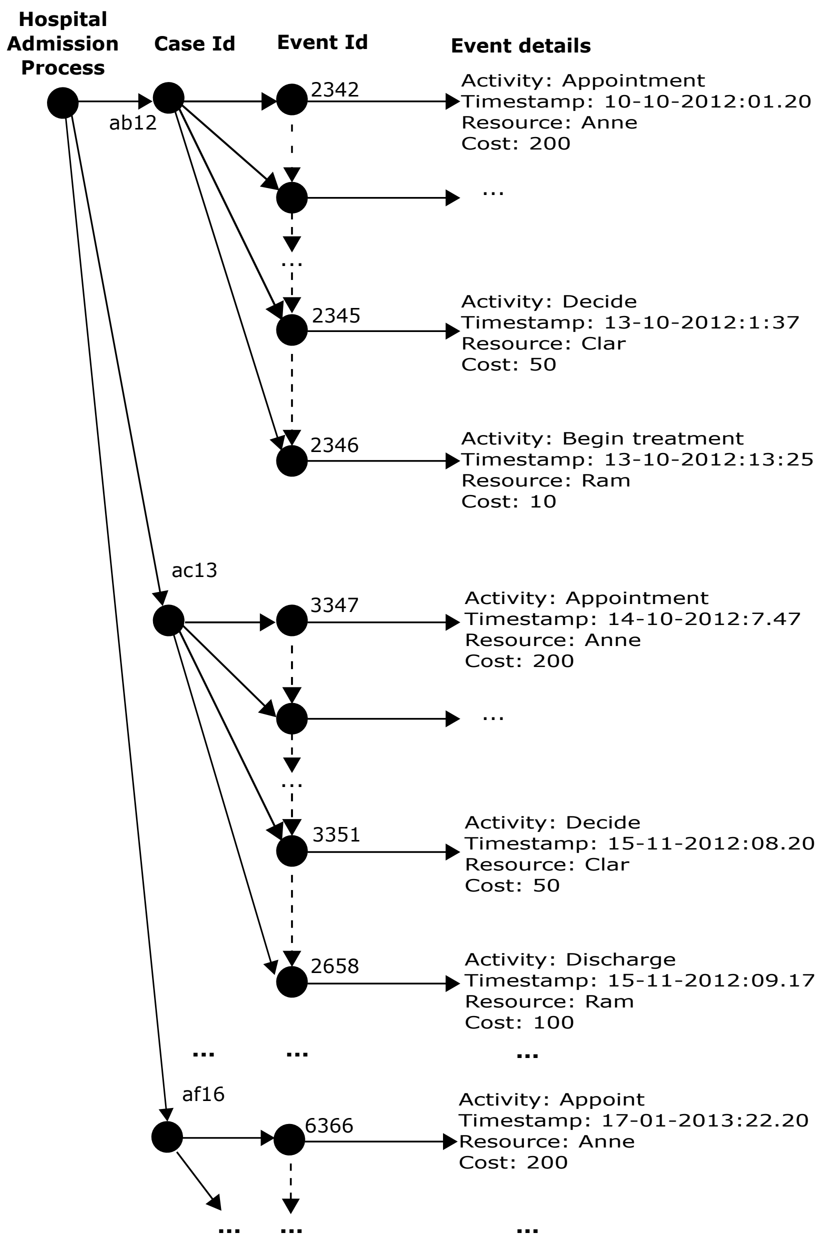


Figure 2.1: Tree structure of process log (taken from Van Der Aalst (2011) and adopted according to the context of the thesis).

– ***Activity*** assigns a readable name for every event of a case. For example, event 2346 of case ab12 and event of 3347 of case ac13 points to an activity named `Appointment`. But activity `Appointment` will be carried out separately for both the cases.

– ***Resources*** identify the individuals or machines who are assigned and responsible for executing a specific activity. For example, Ram is assigned as a resource for executing the activity `Discharge` related to all cases.

– ***Timestamps*** record the duration between the start and end of a particular activity.

- The expenditure incurred while executing a specific activity is recorded in the ***cost*** field.
- Data objects related with the process are recorded in the ***data*** field. Typical data objects are messages, files, documents, guards, videos, voice, and conditions.

We now formalize the various notations related to event log,

Definition 2.8. (Event and attribute) Let $\mathcal{E}$ be the set of all event identifiers and AN be the set of all attributes. For an attribute $s \in AN$ let $\mathcal{X}_x$ is a universal set (consists of set of all possible values of x). Given $\mathcal{E}$ and an attribute $x \in AN$, $\#_x(e) = \perp$ is the value for all attributes x not defined in e and $\#_x : \mathcal{E} \rightarrow \mathcal{X}_x \cup \{\perp\}$ denotes value of attribute x for any event $e \in \mathcal{E}$.

Definition 2.9. (Classifier)
A classifier is a function that maps an event into a representative name used for particular type of analysis. For any event $e \in \mathcal{E}$ let $\bar{e}$ be the name of the event. For example, if events are identified by their activity names, then $\bar{e} = \#_{activity}(e)$.

For any given event e, we use standard attributes such as activity, resource and time. Let $\mathcal{A}$ be the set of activities, $\mathcal{T}$ be the time domain, $\mathcal{R}$ be the set of resources, and TT be set of transaction types, then,

- $\#_{activity}(e) \in \mathcal{A}$ signifies the activity associated with event e.
- $\#_{resource}(e) \in \mathcal{R}$ indicates the resource performing the event e.
- $\#_{time}(e) \in \mathcal{R}$ indicates the timestamp of event e.
- $\#_{trans}(e) \in TT$ indicates the transaction type associated with the event e. Transaction type attribute refers to activity life-cycle phases such as schedule, start, suspend, complete, etc.

Definition 2.10. (Case, Trace and Event log)
Let $\mathcal{C}$ is the set of all case identifiers and AN be the set of all attributes. For any given case $c \in \mathcal{C}$ and attribute $x \in AN$, $\#_x : \mathcal{C} \rightarrow \mathcal{X}_x \cup \{\perp\}$ indicates value of attribute x for case c.
Trace (a mandatory attribute of a case) represents a finite sequence of events $t \in \mathcal{E}^*$ such that no two event in trace are same, i.e., for $1 \leq i < j \leq |t|, t(i) \neq t(j)$

If activity name is used as a classifier, then a trace corresponds to sequence of activities. Such scenario results in more than one cases with same activity sequence. Therefore, an event log is a multi-set of traces.

Definition 2.11. (Simple event log)
Let $\mathcal{A}$ be a set of activity names. A simple trace σ is a sequence of activities, i.e., $\sigma \in \mathcal{A}^*$. A simple event log $\mathcal{L}$ is a multi-set of traces over $\mathcal{A}$, i.e., $\mathcal{L} \in \mathbb{B}(\mathcal{A}^*)$.

All cases in event log $\mathcal{L}$ can be converted into sequences of activity names using the classifier, $\#_{activity}(e)$. Applying this classifier to the cases shown in table 2.1, then we obtain the following simple event log,

$\mathcal{L}$ =[⟨ `Appointment, Admit patient, Check history, Decide, Begin treatment` ⟩, ⟨ `Appointment, Out patient, Check history, Decide, Discharge`⟩, ⟨ `Appointment, Admit patient, Check history, Decide, Re-examine, Out patient, Check history, Decide, Discharge` ⟩, ⟨ `Appointment, Check history, Admit patient, Decide, Discharge`⟩, ⟨ `Appointment, Out patient, Check history, Decide, Re-initiate, Admit patient, Check history, Decide, Begin treatment` ⟩, ..]

Projection using resource classifier can be used in mining organizational models, social networks, etc. By choosing resource classifier $\#_{resource}(e)$, the sample event log results as,

$\mathcal{L}$ =[⟨ Anne, Jhon, Thomas, Clar, Ram ⟩, ⟨ Anne, Wil, Thomas, Clar, Ram⟩, ⟨ Anne, John, Thomas, Clar, Steven, Wil, Thomas, Clar, Ram ⟩, ⟨ Anne, Thomas, John, Clar, Ram⟩, ⟨ Anne, Wil, Thomas, Clar, Steven, John, Thomas, Clar, Ram ⟩, ..]

2.2.1 Control-flow: causal relationships

The least necessity for process mining is that any event can be related to both a case and an activity and that events within a case are ordered. Therefore, the case id and activity columns in process log are necessary for process mining. Based on the type of analysis, a subset of information available in event log is considered. For example, activity column is the bare minimum requirement for discovering the control-flow perspective of a process. Which shows the causal relationships between activities in

Case id	Trace
xx12	⟨ a, b, d, e, g ⟩
xx13	⟨ a, c, d, e, h ⟩
xx14	⟨ a, b, d, e, f, c, d, e, h ⟩
xx15	⟨ a, d, b, e, h ⟩
xx16	⟨ a, c, d, e, f, b, d, e, g, l, m ⟩
...	...

Table 2.2: control-flow traces of hospital admission process.

a process. Control flow traces of hospital admission process is shown in Table 2.2. Here, each case is represented by a chain of activities which results as a trace. For simplicity, the activity names are mapped to alphabet labels.

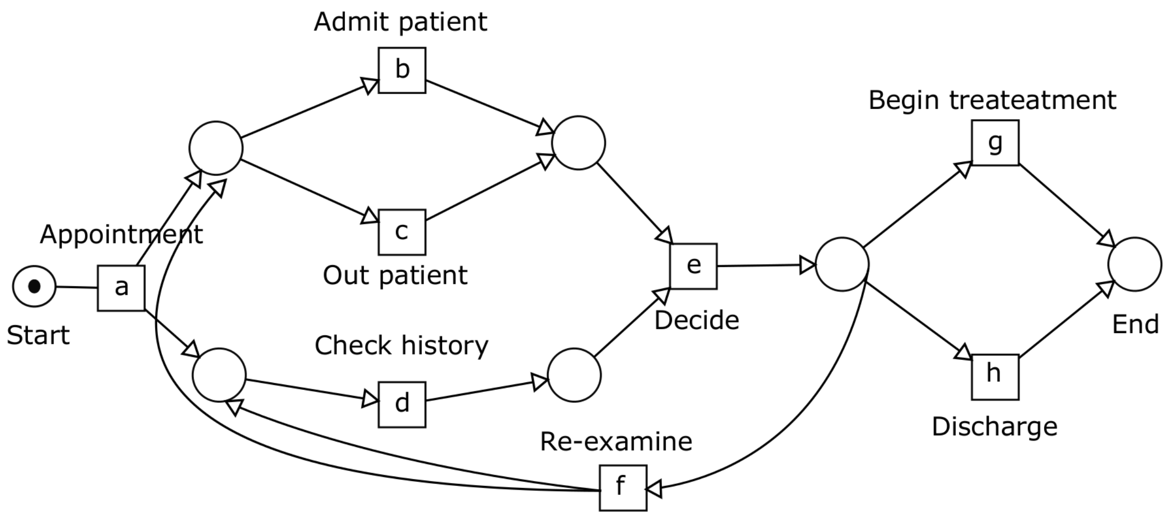


Figure 2.2: Petri-net model of hospital admission process.

Control-flow model depicting the causal relationship between activities of hospital admission process is shown in fig. 2.2. It is built using petri net (a petri net is a triplet $N = (P, T, F)$ where P is a finite set of places, T is a finite set of transitions such that $P \cap T = \phi$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs, called the flow relation.) notations. According to control flow process model shown in fig. 2.2, a patient has to take an *appointment* (a) and visit the hospital, based on the condition of patient, the case is considered as *out patient* (c) or *admit patient* (b). Concurrently, the *medical history*(d) of the patient is verified. On the basis of verification outcome, the *decision* (e) is taken either to *admit* (g) or *discharge* (h) the patient. If the case still needs further evaluation, case is *re-examined* (`f`) and the process is carried out allover again.

2.3 Representing and Storing Event Log

For analyzing event log using process mining methods, event logs should be based on typical process meta-model such as Mining eXtensible Markup Language (MXML) format. For addressing the practical limitations in MXML (Van Der Aalst, 2011), IEEE task force on process mining has proposed and adopted a new process meta model named eXtensible Event Streams (Verbeek et al., 2010).

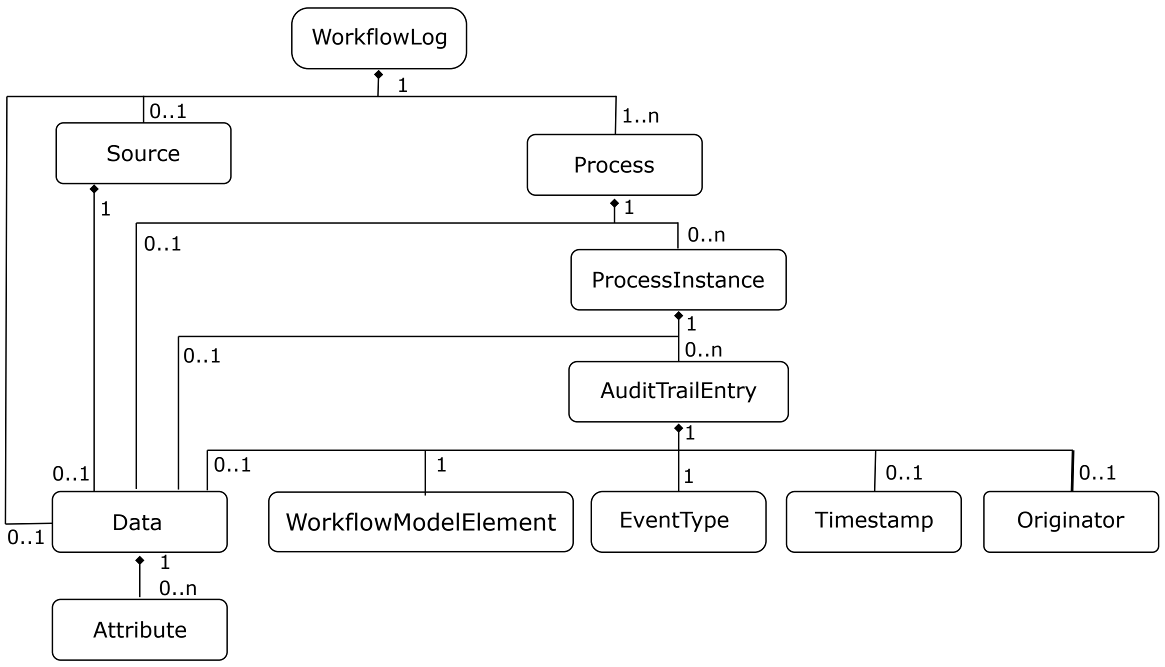


Figure 2.3: Event log structure in MXML format.

2.3.1 Mining eXtensible Markup Language

Unified Modeling Language (UML) class diagram of MXML has shown in fig. 2.3. Fragment of event log related to hospital admission process in MXML format has been shown in the listing 2.1.

The root element in MXML format is `WorkflowLog`. Which contains the process information related one or more processes, it can optionally can contain `Source` element, which signifies the system from which the particular log has been obtained.

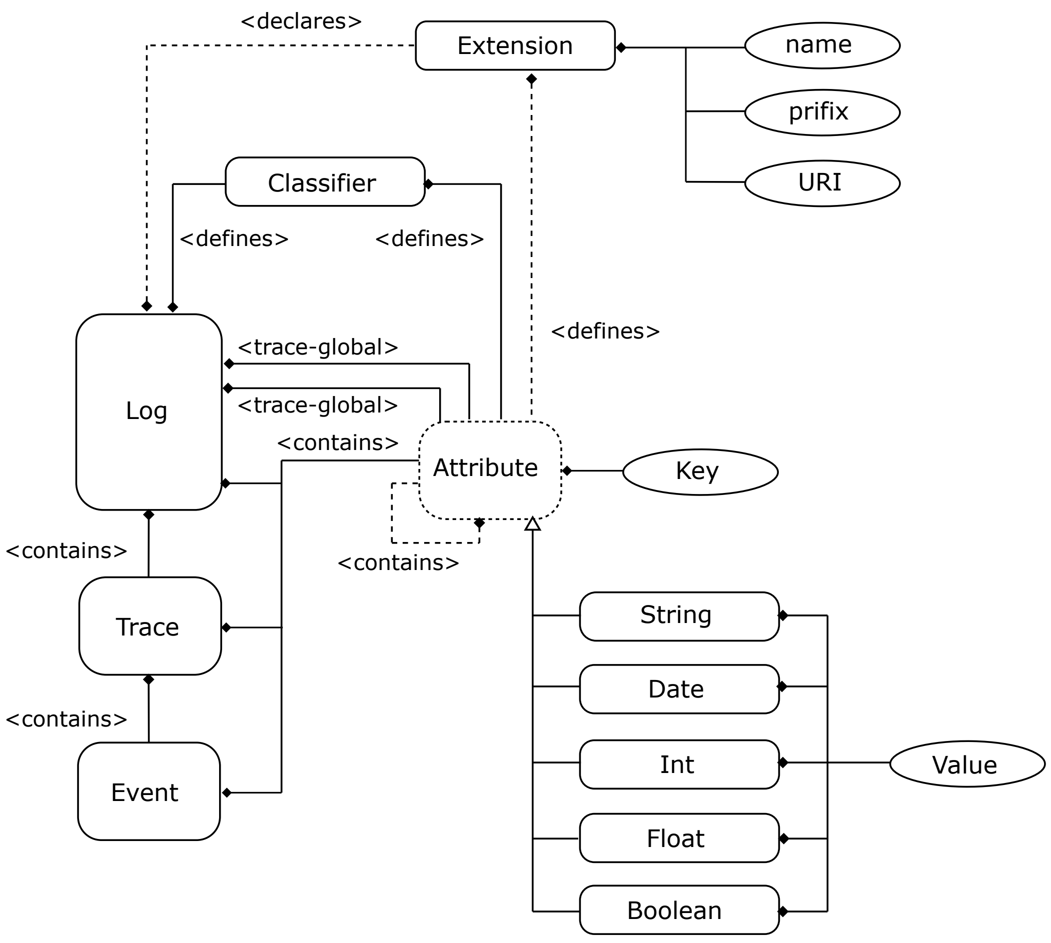


Figure 2.4: Event log structure in XES format.

Each `Process` in `WorkflowLog` consists of any number of `ProcessInstances`. Each `ProcessInstance` is made-up of ordered collection of events termed as `AuditTrail Entry`.

`WorkflowModelElement` and `EventType` are the mandatory constituent parts of `AuditTrailEntries`. It can optionally contain additional information such as `Time stamp` and `Originator`. All elements in MXML format have optional `Data` element for storing additional information as `Attributes` in the form of key and value pairs.

MXML suffers from a few limitations such as,

- It can only be used for recording the events originating from well-structured processes and workflow environments.
- It cannot be employed for capturing the events emanating from non-workflow systems.
- It doesn't support extensibility options to incorporate additional knowledge.

2.3.2 eXtensible Event Streams

Until recently, MXML and its variant such as Semantically Annotated Mining eXtensible Markup Language (SA-MXML) are the de-facto standards for storing and exchanging event logs in digital format. Based on many practical limitations with MXML (and SA-MXML), the XES format has been accepted as a standard event log format. XES has been made less restrictive and truly extensible.

General structure of XES metamodel is shown in fig.2.4. An XES document contains one event log with any number of cases and attributes which can be nested. XES permits the usage of five basic data types, namely, Boolean, Integer, String, Date, and Float for the standard built-in data types of XML xs:boolean, xs:int, xs:string, xs:dateTime, and xs: float respectively. The attribute that is mandatory should be declared as global. Listing 2.2 shows the part of event log related to hospital admission process in XES format.

XES permits the declaration of five standard extensions. Through extensions semantics of attributes are specified.

- *Concept:* defines the name attribute for traces and events. For traces, the attribute typically represents distinct identifier for the case. For events, the attribute typically represents the activity name.
- *Time:* defines the timestamp attribute for events.
- *Organization:* defines three standard attributes for events: resource, role and group. The resource attribute refers to the resource that triggered or executed the event. The role and group attributes characterize the (required) capabilities of the resource and the resource's position in the organization.
- *Lifecycle:* defines the transition attribute for events, possible values of this attribute are schedule, start, complete, auto, skip, etc.
- *Semantic:* defines the model reference attribute for all elements in the log. This is used for pointing to concepts in the ontology. For example, if there is any ontology explaining various classes of memberships, for example, Platinum, Gold, and Silver. Using semantic extension, any given trace can refer to the suitable element in the ontology for classifying the memberships.

Listing 2.1: Event log of hospital admission process in MXML format.

```
<WorkflowLog ...>
  <Source program="Hospital process"/>
  <Process ...>
    <ProcessInstance id="xx12">
      <AuditTrailEntry>
        <Data>
          <Attribute name="Costs">200</Attribute>
        </Data>
        <WorkflowModelElement>Appointment</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>10-10-2012T01:20</Timestamp>
        <Originator>Anne</Originator>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <Attribute name="Costs">100</Attribute>
        </Data>
        <WorkflowModelElement>Admit patient</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>11-10-2012T07:24</Timestamp>
        <Originator>John</Originator>
      </AuditTrailEntry>
      <AuditTrailEntry>
      <Data>
        <Attribute name="Costs">400</Attribute>
        </Data>
          <WorkflowModelElement>Check history</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>12-10-2012T08:24</Timestamp>
        <Originator>Thomas</Originator>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <Attribute name="Costs">50</Attribute>
        </Data>
        <WorkflowModelElement>Decide</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>13-10-2012T12:24</Timestamp>
        <Originator>Clar</Originator>
      </AuditTrailEntry>
      <AuditTrailEntry>
        <Data>
          <Attribute name="Costs">10</Attribute>
        </Data>
        <WorkflowModelElement>Begin treatment</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>13-10-2012T13:25</Timestamp>
        <Originator>Ram</Originator>
        </AuditTrailEntry>
    </ProcessInstance>
    <ProcessInstance id="xx13">
      <AuditTrailEntry>
        <Data>
          <Attribute name="Costs">50</Attribute>
        </Data>
        <WorkflowModelElement>Appointment</WorkflowModelElement>
        <EventType>complete</EventType>
        <Timestamp>14-10-2012T01:29</Timestamp>
        <Originator>Anne</Originator>
      </AuditTrailEntry>
      ...
    </ProcessInstance>
    ...
  </Process>
</WorkflowLog>
```

Listing 2.2: Event log of hospital admission process in XES format.

```
<?log xes.version="1.0" xes.features="nested-attributes">
  <extension name="Concept" prefix="concept" uri="http://.../concept.xesext"/>
  <extension name="Time" prefix="time" uri="http://.../time.xesext"/>
  <extension name="Organizational" prefix="org" uri="http://.../org.xesext"/>
  <global scope="trace">
    <string key="concept:name" value="name"/>
  </global>
  <global scope="event">
    <date key="time:timestamp" value="2012-09-16"/>
    <string key="concept:name" value="name"/>
    <string key="org:resource" value="resource"/>
  </global>
  <classifier name="Activity" keys="concept:name"/>
  <classifier name="Resource" keys="org:resource"/>
  <classifier name="Both" keys="concept:name org:resource"/>
  <trace>
    <string key="concept:name" value="xx12"/>
    <event>
      <string key="concept:name" value="Appointment"/>
      <string key="org:resource" value="Anne"/>
      <date key="time:timestamp" value="10-10-2012T01:20"/>
      <string key="Event_ID" value="2342"/>
      <string key="Costs" value="200"/>
    </event>
    <event>
      <string key="concept:name" value="Admit patient"/>
      <string key="org:resource" value="John"/>
      <date key="time:timestamp" value="11-10-2012T07:24"/>
      <string key="Event_ID" value="2343"/>
      <string key="Costs" value="100"/>
    </event>
    <event>
      <string key="concept:name" value="Check history"/>
      <string key="org:resource" value="Thomas"/>
      <date key="time:timestamp" value="12-10-2012T08:24"/>
      <string key="Event_ID" value="2344"/>
      <string key="Costs" value="50"/>
    </event>
    <event>
      <string key="concept:name" value="Decide"/>
      <string key="org:resource" value="Clar"/>
      <date key="time:timestamp" value="13-10-2012T12:24"/>
      <string key="Event_ID" value="2345"/>
      <string key="Costs" value="50"/>
    </event>
    <event>
      <string key="concept:name" value="Begin treatment"/>
      <string key="org:resource" value="Ram"/>
      <date key="time:timestamp" value="13-10-2012T13:25"/>
      <string key="Event_ID" value="2346"/>
      <string key="Costs" value="10"/>
    </event>
  </trace>
  <trace>
    <string key="concept:name" value="xx13"/>
    <event>
      <string key="concept:name" value="Appointment"/>
      <string key="org:resource" value="Anne"/>
      <date key="time:timestamp" value="14-10-2012T01:29"/>
      <string key="Event_ID" value="3347"/>
      <string key="Costs" value="200"/>
    </event>
    ...
  </trace>
  ...
</log>
```

2.4 Process Mining Framework: ProM

In the early days of process mining research, a numerous ad-hoc process mining tools were created which were limited to the only subset of process mining algorithms. Examples for the ad-hoc tools are, Enhanced Mining Tool (EMiT) (Van Dongen and Van Der Aalst, 2004), Mining Social Networks (MinSon) (Van Der Aalst and Song, 2004), Mining Module (MiMO) (Van Der Aalst et al., 2004), and Rule of thumb (Weijters and Van Der Aalst, 2003). In 2004, the development of ProM was started to combine all the available process mining ad-hoc and future implementations.

2.5 Artificial Event Log Synthesis

Process Mining algorithms are target applications for real-world event logs. A majority of times, real-world event logs are not the best data to evaluate a process mining algorithm during the development phase. The main reason for this is real-world event logs may be far from complete and contain outliers, which can pose difficulty in determining whether an incorrect result is due to input log or mining algorithm. It is necessary to retain control over the test data for evaluating the quality of the new mining algorithm.

An effective way for testing a newly developed process mining algorithm is to use artificially created event log. Artificially created data offers the complete control over the various characteristic of event data. Once the algorithm can generate the expected results using artificial data, the event logs generated by real-world processes become target application for process mining algorithms.

A preferable way for synthesizing artificial event log is simulating process models. This implies that an executable process model is created under controlled conditions, allowing event log to record the predefined events. A variant of Petri-nets called Colored Petri Nets (CPN) (Jensen and Kristensen, 2009) can be used for accomplishing this purpose. CPN in conjunction with CPNXES library (Michael, 2011) allows a designer to keenly control the semantics of execution, by focusing on the desired characteristics.

Modeling, execution, and analysis of CPN is supported by CPN tools (Jensen and Kristensen, 2009). Following are the two steps required for synthesizing CPN using

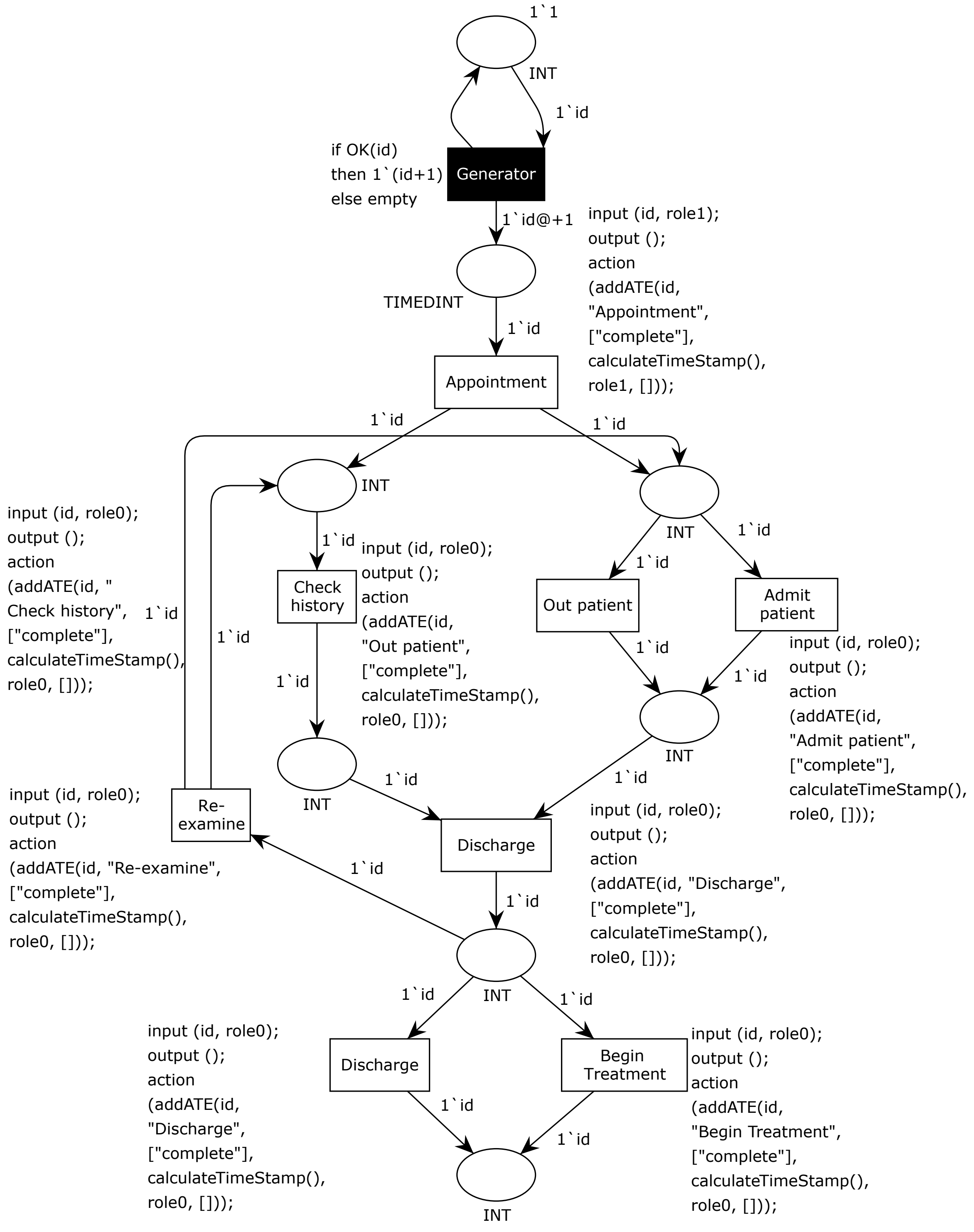


Figure 2.5: CPN model of hospital admission process.

CPN Tools.

- Creating CPN net to call functions for generating events for every case executed. It consists of a simulated process model, along with the functions for input, action, and output on transitions.

- Use ProMimport plugin (Van Der Aalst et al., 2007a) in ProM framework to group the events in the event log for independent cases into generating single XES or MXML file.

Equipping a CPN model for generating event logs is a straightforward process. From the CPN model, there needs to call two different logging functions. The first function allows to create a new case with case id and the second function generates an event on specific case. The case can further be explained by name, resource, cost, etc.

The most time-consuming step in generating artificial event log is creating CPN process model. Once it is created, it is straightforward to extend it to logging facility. One such example of CPN model with logging facility has been shown in fig. 2.5. Which represents CPN model of hospital admission process.

Chapter 3

Literature Survey

There are several contributions done by the researchers all over the world which has helped us in identifying the research gaps, and addressing them by proposing the suitable solutions. This chapter presents the set of scientific literature we have referred in this thesis. Literature related to three diverse categories of research issues in process mining are discussed in the upcoming sections, at *first* we discuss the study related to non-stationary learning problem called concept drift, *then* we present the extensive study on the set of available control-flow process modeling notations and algorithms, and *finally* the concept of Spaghetti and Lasagna process has been discussed.

3.1 Process Mining

Process mining emerged in the last decade (Van der Aalst et al., 2003; Van Der Aalst et al., 2007a). However, the roots date back about half a century. For example, Nerode (1958) presented an approach to synthesize finite-state machines from example traces, Petri (1962) introduced the first modeling language adequately capturing concurrency, and Gold (1967) was the first to systematically explore different notions of learnability. When data mining started to flourish in the nineties, little attention was given to processes. Moreover, only recently event logs have become omnipresent thus enabling end-to-end process discovery. Since the first survey on process mining (Van der Aalst et al., 2003), progress has been spectacular.

Process mining techniques have become mature and supported by various tools. Moreover, whereas initially the primary focus was on process discovery, the process mining spectrum has broadened markedly. For instance, conformance checking, multi-perspective process mining, and operational support have become integral parts of ProM, one of the leading process mining tools.

3.1.1 Overview of Process Mining

Process mining is a relatively young research discipline that sits between computational intelligence and data mining on the one hand, and process modeling and analysis on the other hand. The idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's (information) systems. Process mining includes (automated) process discovery (i.e., extracting process models from an event log), conformance checking (i.e., monitoring deviations by comparing model and log), social network/ organizational mining, automated construction of simulation models, model extension, model repair, case prediction, and history-based recommendations.

Process mining provides an important bridge between data mining and business process modeling and analysis. Under the Business Intelligence (BI) umbrella many buzzwords have been introduced to refer to rather simple reporting and dashboard tools. Business Activity Monitoring (BAM) refers to technologies enabling the real-time monitoring of business processes. Complex Event Processing (CEP) refers to technologies to process large amounts of events, utilizing them to monitor, steer and optimize the business in real time. Corporate Performance Management (CPM) is another buzzword for measuring the performance of a process or organization. Also related are management approaches such as Continuous Process Improvement (CPI), Business Process Improvement (BPI), Total Quality Management (TQM), and Six Sigma. These approaches have in common that processes are put "under a microscope" to see whether further improvements are possible. Process mining is an enabling technology for CPM, BPI, TQM, Six Sigma, and the like.

3.2 Concept Drift

The word concept drift was coined by Schlimmer and Granger Jr (1986) in the article *"Incremental learning from noisy data"*, and since then, the concept drift phenomenon is widely considered in all research disciplines involving data analysis. Concept drift in many data research disciplines is referred to as a *change in properties of the target variable which the model is trying to predict over time in unforeseen ways.* This causes the predictions become less precise over the time. Hence, it makes concept drift as a non-stationary learning problem (Hand et al., 2006). The major concentration is on designing the efficient learning algorithms that can adapt to data changes over time, for example, variations in the distributions of numerical, categorical and nominal variables.

Learning algorithms usually required to operate in changing environments, which is shifting unexpectedly. One general property of these algorithms is their ability to include new data. If the data producing process is not stringently stationary, the underlying concept, which we are predicting (for example, interests of a user reading news), may be shifting over time. The capacity to adjust to such concept drift can be seen as a natural expansion for data analysis algorithm.

Concept drift phenomenon has been extensively under consideration for research in various data analysis discipline from the past two decades (Widmer and Kubat, 1996; Tsymbal, 2004; Schlimmer and Granger, 1986). Concept drift is known as *co-variate shift* in machine learning (Blitzer et al., 2008; Jiang and Zhai, 2007), *temporal evolution* in information retrieval (Wu et al., 2005), *dynamic environment* in artificial intelligence and robotics (Luo et al., 2007; Yang and Yao, 2008), and *non-stationary learning problem* in time series analysis (Hand et al., 2006). Unfortunately, concept drift phenomenon is not addressed in process mining (Van Der Aalst et al., 2012).

3.2.1 General approach for handling concept drift

Effective learning in environments with concept drift requires a learning algorithm that can detect context changes without being explicitly informed about them, can quickly recover from a context change and adjust its hypothesis to a new concept, and can make use of previous experience in situation where old and corresponding concepts reappear.

One possible approach is shown in fig. 3.1. As the context is known to vary in time, the learner trusts only the latest examples. This set is referred to as window. Examples are added to window as they arrive, and oldest examples are deleted from it. Both of these actions trigger modifications to current concept hypothesis to keep

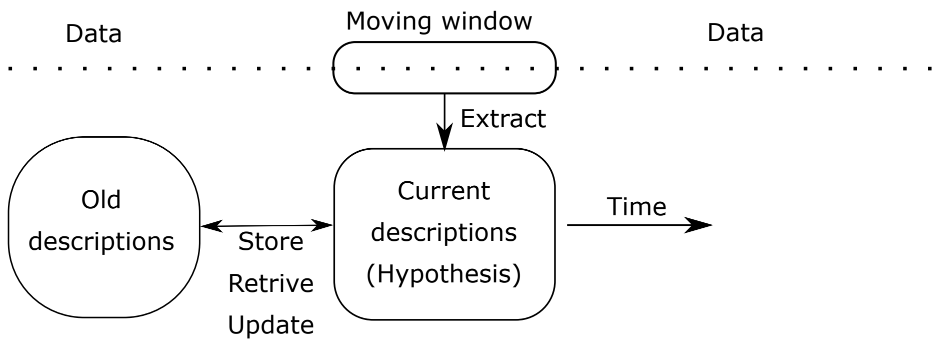


Figure 3.1: Generalized approach for addressing concept drift.

it consistent with the examples in the window.

Learner maintains a store of concept descriptions or hypothesis pertaining to previously encountered contexts. This is indicated in the lower left part of the fig. 3.1. When learner suspects a concept change, it will examine the potential of previously stored concepts to provide better classifications. Based on the result, the system may either replace the current concept descriptions with best stored descriptions, or start developing an entirely new description.

For efficiently handling concept drift, the following set of ideas are necessary.

- Operators that modify the concept description in reaction to changes in content of window.
- Ability to decide when and which concepts should be deleted from set of old descriptions.
- Strategy that maintains the store of current and old descriptions.
- Strategy to asses the relative merits of stored and current hypothesis.

3.2.2 Concept drift in process mining

In process mining, the task of handling concept drift is to discover the changes and update the process model to reflect the latest state of the process. Specifically, handling concept drift seeks to analyze changes in the complex artifacts such as models describing *choices*, *concurrency*, *loops* and *cancellation*. However, the experiences from other research disciplines can be adopted to deal with the concept drift in process mining. But, the nature of process changes and structure of process models pose new challenges and demands the development of suitable techniques.

A Clustering and classification

There has been some attempts to find the versions of a process by applying clustering and classification methods of data mining (Song et al., 2009; Luengo and Sepúlveda, 2012; Bose and Van Der Aalst, 2009b; Bezerra et al., 2009). But, finding different versions of the process does not consider the type, pattern, and perspective of concept drift. Hence there is a need for research to find better solutions to handle concept drift.

B Use of flexible execution patterns

Currently, in the field of process flexibility, configuring the workflow system has become adaptive. A collection of change patterns related to control-flow (Russell et al., 2006), data (Russell et al., 2004a), and resource (Russell et al., 2004b) perspectives have been described. Schonenberg et al. (2008) and Regev et al. (2006) have provided the detailed classification of several flexibility and adaptability methods. Processes executing in information systems can be benefited from these flexible patterns, at the same time, the same method cannot handle the unforeseen changes in the process.

C Use of change logs

Gunther et al. (2008) proposed a method of finding process changes with the help of change logs recorded in the information systems. As of now, very few systems are capable of generating change logs. Hence, there is a necessity to handle concept drift based solely on event log input.

D Feature set extraction and processing

Bose et al. (2011) proposed a method for addressing concept drift in the article *"Handling concept drift in process mining"*. They proposed a technique for handling control-flow concept-drift by evaluating hypothesis on feature values derived from the event log. Concept drift is said to be detected when a continuous dip in significance probability is observed. The drawbacks of this method are,

- Infeasible for processing large event logs (proposed feature set results in a large number of values),

- The dimensions of features increase as the number of activities in the process increases.
- Extracting feature values by processing individual trace is time-consuming.

These issues are addressed in our proposed method by making use of consensus trace produced from aligning traces. Consensus trace decreases the size of the event log and in turn, eliminates the need for processing every individual trace for extracting features related to control-flow.

E Theory of abstract interpretation

Carmona and Gavalda (2012) presented an *online* method for handling concept drift. They proposed a multi-stage technique that uses the theory of abstract interpretation of control-flow in the form of polyhedra. Successive traces were evaluated to check whether they reside inside the polyhedra. If a sample resides inside, it is deemed to indicate the unchanged process. If considerable samples reside outside the polyhedra, it signifies process change. Following are the major drawbacks of this method,

- This method is valid only for change detection.
- It does not identify the existence of more than one change, as soon as it encounters the first change it reports and gets terminated even if there are several changes in the process.

The method proposed for handling concept drift in this thesis not only detects but also localizes all control-flow concept drifts in single run of the algorithm.

3.3 Characterization of Concept Drift

Characterization helps in designing and developing an appropriate solution for addressing the concept drift. This section presents the various aspects (shown in fig. 3.2) to be considered while characterizing the concept drift in a process.

3.3.1 Perspectives

Perspectives facilitate to witness the process from different viewpoints (Van Der Aalst, 2011). In the context of process mining, concept drift can occur in control-flow, data,

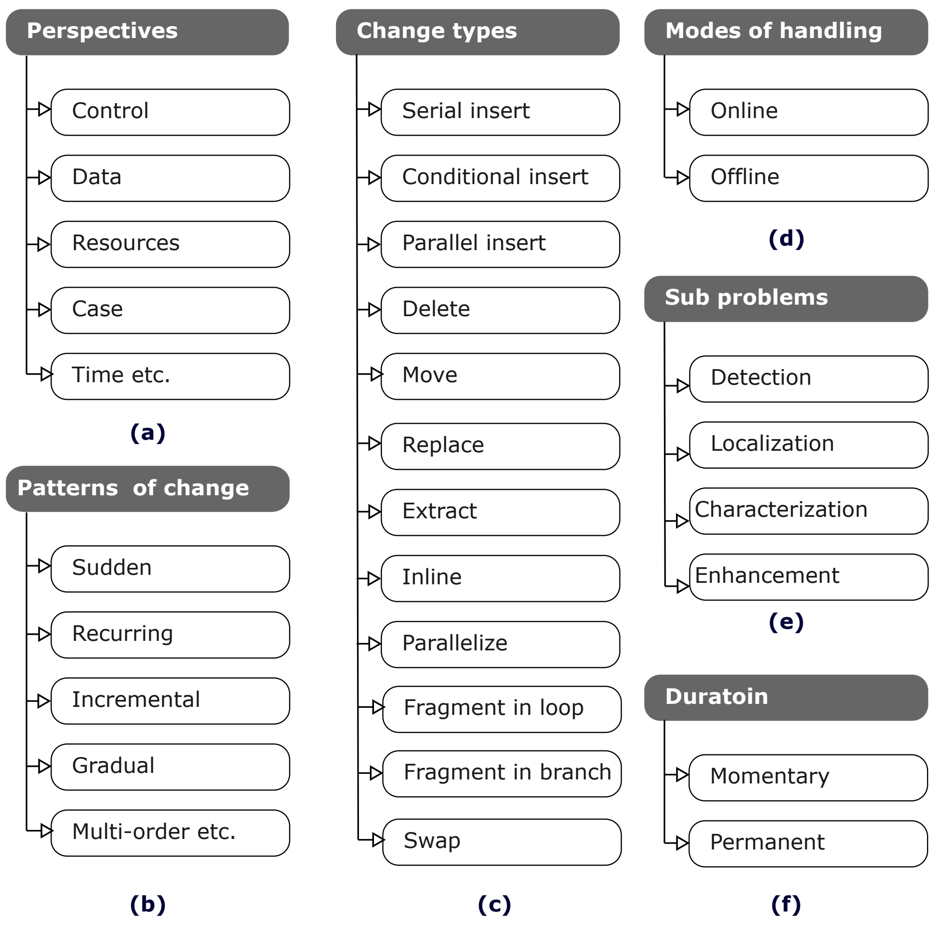


Figure 3.2: Dimensionality of concept drift phenomenon for characterizing changes.

resource, organizational, time and case perspectives.

- Concept drift in control-flow perspective refers to the changes in casual relationship between activities of process. According to Weber et al. (2007), frequently observable modifications in control-flow can be classified into change types such as (fig. 3.2c),

  - *Serial insert:* an activity/fragment (fragment is a set of connected activities) is inserted between two directly connected activities.
  - *Conditional insert:* an activity/fragment added with an additional condition.
  - *Parallel insert:* adding an activity/fragment in parallel to other activities/fragments.
  - *Delete:* permanently removing an activity/fragment.
  - *Move:* Transferring an activity/fragment to another place.

  - *Extract:* substituting a sub process by a "reference activity".
  - *Replace:* substituting a fragment/activity with another fragment/activity.
  - *In-line:* exact opposite of the extract operation.
  - *Parallelize:* Parallelizing sequential fragments/activities.
  - *Fragment in branch:* making process fragment as one of many possible choices during the process execution.
  - *Fragment in loop:* arranging a process fragment in a loop, so that it can be executed multiple times.

- Concept drift in *data perspective* indicate changes in data objects, variables and conditions that guide and control the execution.
- Concept drift in *resource perspective* signifies changes in assets, roles and their effect on the execution of the process.
- Concept drift in *organizational perspective* signifies changes in typical work patterns, organizational structures and social networks.
- Concept drift in *time perspective* affects frequency, duration, utilization, prediction, and timing of events.
- Concept drift in *case perspective* alters properties, attributes, path and performance.

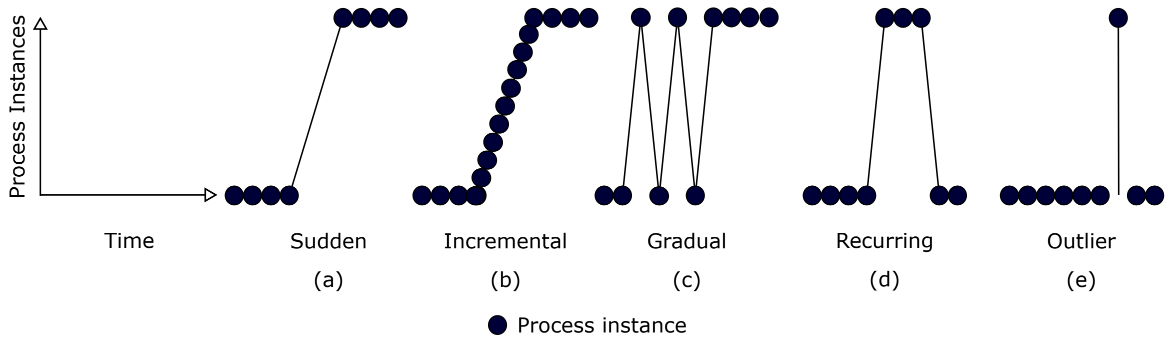


Figure 3.3: Most common concept drift patterns.

Most of adaptive learning techniques implicitly or explicitly assume and specialize in some subset of concept drifts. Many of them assume sudden non-reoccurring drifts. But in reality often mixtures of many types can be observed.

3.3.2 Patterns of concept drift

Sudden, recurring, incremental, gradual, multi-order, and *outliers* (Gama et al., 2014) are some of the most frequently observed patterns of change in a process (listing shown in fig. 3.2b).

- *Sudden* (Bose et al., 2011) concept drift is when there is an abrupt change in the operation of a process (as shown in fig. 3.3a).

  This pattern is typically seen during the time of *emergency response planning, crisis situations* and c*hange of law/legislation.*

- Concept drift is called *incremental* (Tsymbal et al., 2008; Delany et al., 2005), when the final version of the process is obtained by making changes in all the intermediate versions of the process. It is shown in fig. 3.3b.

  The evolution of the banking process from manual to internet banking, and mobile banking is an example of incremental change pattern.

- Process exhibits *gradual* (Stanley, 2003; Widmer and Kubat, 1993) (shown in fig. 3.3c) concept drift when there are two versions (M_1 and M_2) of the same process active at a given point of time. As time passes, the probability of traces from model M_1 decreases and the probability of traces from model M_2 increases. At a certain point in time, process M_1 is replaced by M_2, and traces from process M_1 cannot be observed.

  For example, consider replacing the delivery process of an online shopping website. The orders received before the change of delivery process are delivered using model M_1, and the orders received after the change are shipped according to M_2.

- Concept drift is known as *recurring* (Tsymbal, 2004), when a previously active concept reappears after some time (shown in fig. 3.3d). Recurring pattern can be *periodic* or *non-periodic.*

  For example, a peak in the sales of ice cream is associated with the summer season, but there may be an increase in sales at different times every year depending on increased temperature or some public event. Depending on the ice cream sales, the vendor has to employ different process model to satisfy the demand.

- Momentary change in operation of the process is termed as *Outliers* (shown in fig. 3.3e). Outliers can also be thought as error or noise in the process log.

  For example, the way in which the government agencies carryout operations during the time of natural calamity or disaster. This type of concept drift does not last for long and very rare to observe.

3.3.3 Sub-problems

Following are the issues to be addressed for efficiently handling concept drift in process mining.

1. *Concept drift detection:* The first and foremost issue is to detect that there is some concept drifts in the process. This demands the techniques for detecting concept drift, irrespective of nature and characteristics of changes in the process.

2. *Concept drift localization:* It involves the investigation of the cases that are affected and the time interval during which concept drift occurred.

3. *Concept drift characterization:* Concept drift characterization involves identifying the *perspective* (control-flow, data, organizational, case etc.), *change type,* (insert, delete, etc.) and *pattern of change* (sudden, recurring, incremental etc.).

4. *Change process discovery*: Change process discovery results in the discovery of a new(changed) process. This step in particularity demands the ways and means that exploit and relate the concept drift detection and localization findings with the existing structure of the process.

3.3.4 Online and offline techniques

The techniques for addressing concept drift can be implemented either in *online* or *offline* modes. Online methods detect and localize concept drift in real-time. On the other hand, an offline method does the posterior investigation of the recorded process instances to locate changes in the process. The offline method is generally carried out by processing the cases in the event log in batch mode (Zliobaite, 2010; Tsymbal, 2004).

3.4 Representing Control Flow Perspective

The control flow notations connect activities (i.e., functions, tasks, transitions) in the process through constructs like condition, places, events, connectors, and gateways. The goal of a control flow model is to visualize the order of the activities to be executed. Based on the transitions between activities, execution can be serial, concurrent, optional, and repeated. Process mining offers a plethora of techniques for discovering control-flow model out of event-log. These discovery algorithms are capable of representing control-flow using various notations. This section aims at listing a subset of most widely used process modeling notations.

3.4.1 Process discovery techniques

Process Mining techniques are aimed to deliver a deep understanding of the operational process by using the reservoir of information that exists in within event logs. One of the main driving forces of the success of process mining is certainly the graphical visualization.

Process discovery is one sub-domain of process mining that aims at construction of control-flow from such event logs. Despite the vastness of the process mining research discipline, most of the attention in the process mining literature has been given to control flow discovery methods. In this section, various control flow modeling notations are presented in order to systematically assess the quality of process discovery techniques.

Since the invention of process mining research discipline, probabilistic, machine learning, as well as algorithmic methodologies for visualizing control flow perspective, have been proposed. Development of ProM framework by TU/Eindhoven makes these techniques available as a single tool.

A Initial approaches

The initial methods to process discovery were articulated by Datta (1998), Cook and Wolf (1998), and Agrawal et al. (1998) . Cook and Wolf demonstrated RNet, Ktail and Markov methods concentrating the discovery of control-flow from an algorithmic, probabilistic, and statistical perspectives. Notion of relating process discovery in the ground of workflow management is proposed by Agrawal et al. and Datta.

Mannila and Meek (2000) presented the method for constructing the understandable global view of a set of sequences with the help of component mixture model of global partial orders. This technique cannot deal with concurrency and several additional characteristics in the process mining.

By employing techniques available in data mining and machine learning Schimm (2002) developed the tool named Process Miner. Process Miner was capable of learning process model from the event log. But, this technique failed to qualify as robust, which is a significant prerequisite in order to employ methods in the real-life working environment.

B α algorithm and It's successors

The invention of α algorithm (Van Der Aalst et al., 2004) considered as one of the most important control flow discovery methods in process mining. It extracts the workflow nets (subclass of Petri nets) from event logs. But, the alpha algorithm is sensitive to noise and incomplete event logs. Several algorithms, such as $\alpha+$ (Alves de Medeiros et al., 2004), $\alpha++$ (Wen et al., 2007) and β (Wen et al., 2009) have been proposed to improve the α algorithm to overcome its drawbacks.

HeuristicsMiner proposed by Weijters et al. (2006) extends the α algorithm and considers frequency information on three kinds of relations among activities in an event log: *direct*, *parallel* and *not direct*. It can learn short loops and non-local dependencies, but cannot detect the duplicate events. HeuristicMiner acts robust in a real-life setting.

Günther and Van Der Aalst (2007) have proposed technique called Fuzzy Miner by considering the idea of visualizing control flow behavior in event logs at various levels of abstraction and aggregation. It allows the analysis of less structured or unstructured operational processes. The fuzzy model cannot be converted to a traditional petri net model. It limits the relative assessment to other control flow discovery techniques.

Table 3.1: Control-flow process discovery algorithms.

Approach	***Author and Year***	***Foundation***	***Prom plug-in***	***Output process model***	***Typical Drawbacks***				
					Outliers	***Splits and Joins***	***Hidden activities***	***Duplicate activities***	***Noise***
Genralized Directed Acyclic Graph	Agrawal et al. (1998)	Algorithmic approach	No	Directed acyclic graph			✓		✓
B-F(fc) Algorithm	Datta (1998)	Algorithmic and Probabilistic approach	No	Process activity graph	✓				✓
Markov, Ktail, and Rnet	Cook and Wolf (1998)	statistical and algorithmic approach	No	Finite state machines	✓		✓		✓
Partial orders-Global	Mannila and Meek (2000)	Algorithmic and probabilistic approach	Yes	Partial orders	✓				
Process Miner	Schimm (2002)	Clustering and classification based	No	Generic linear process model	✓				✓
α and $\alpha+$	Schimm (2002)	Algorithmic approach	Yes	Workflow nets					
Splitpar	Schimm (2004)	Algorithmic approach	No	Workflow nets			✓	✓	✓
Multi-phase Miner	Van Dongen and Van Der Aalst (2005)	Algorithmic approach	Yes	Event driven process chain and Petri nets					
Workflow Miner	Gaaloul et al. (2005)	Algorithmic approach	Yes	Workflow nets	✓	✓	✓		✓
HeuristicsMiner	Weijters et al. (2006)	Algorithmic approach	Yes	Causal Nets	✓		✓	✓	✓
Disjunctive workflow model	Greco et al. (2006)	Clustering and classification based	Yes	disjunctive workflow schema	✓			✓	
Rule-based approach	Măruşter et al. (2006)	Classification based	No	Petri nets					✓
Genetic Miner	De Medeiros et al. (2007)	Genetic algorithmic approach	Yes	Petri nets and Workflow nets	✓		✓	✓	✓
Fuzzy Miner	Günther and Van Der Aalst (2007)	Clustering and algorithmic approach	Yes	Fuzzy Net	✓				✓
$\alpha++$	Ferreira and Ferreira (2006)	Algorithmic approach	Yes	Petri nets				✓	
DecMiner	Lamma et al. (2007)	Inductive logic programing	Yes	DecSerFlow					
AWS Mining	Greco et al. (2008)	Clustering based	Yes	Petri nets	✓				✓
AGNEsMiner	Goedertier et al. (2009)	Classification based	Yes	Petri nets	✓	✓	✓	✓	✓
β	Wen et al. (2009)	Genetic algorithmic approach	Yes	Petri nets	✓				
Enhanced Workflow Miner	Folino et al. (2009)	Genetic algorithmic approach	Yes	Petri nets	✓	✓	✓	✓	✓
EM-approach	Ferreira and Gillblad (2009)	Inductive logic programing	No	Workflow nets					✓
ILP Parikh Miner	Van derWerf et al. (2009)	Genetic algorithmic approach	Yes	Petrinets			✓		
FSM Petrify Miner	Van Der Aalst et al. (2010)	Classification based	Yes	Petri nets in Finite state format	✓	✓	✓	✓	✓
FSM Genet Miner	Carmona et al. (2010)	Classification based	No	Finite state machine		✓			
Inductive Miner	Leemans et al. (2013a)	Classification based	Yes	Petrinets					
Multiple trace alignments	Bose and Van Der Aalst (2010)	Clustering based	Yes	Alignments					

C Application of machine learning techniques

There has been several attempts to use the machine learning techniques for discovering control flow perspective. The majority of methods are based on the application of classification techniques. Măruşter et al. (2006) were the first to investigate the use of rule induction to calculate the dependency relations among activities.

Use of Integer Linear Programming (ILP) learning and partial-order planning techniques to discover process model is demonstrated by Ferreira and Ferreira (2006). In this method, a process model is created by repeatedly combining planning and learning, and the model is represented as case data preconditions and effects. The contribution of this work is included in BPM life cycle of process execution, generation, learning, and re-planning.

De Medeiros et al. (2007) demonstrated the application of a genetic algorithm for process discovery. Parameters (namely fitness and precision) used in its implementation enable genetic algorithm to discover suitable control flow model. Further, the algorithm confirms a robustness by using the technique of post arc running step.

AGNEsMiner (Goedertier et al., 2009) extracts Petri net models from event logs (it models the process discovery as first-order classification problem of events). It uses multi-relational classification learner called Tilde (Ferreira and Ferreira, 2006). Given any event log, AGNEsMiner can discover the conditions that govern the process execution.

With an idea of clustering the patterns which shares similar behavior, an hierarchical and iterative procedure (Greco et al., 2008) that refines the process model has been proposed. This approach improved the traditional discovery techniques by guaranteeing the total compliance with event log (by producing the abstraction taxonomy of process models). It permits the inspection of behavior in the event log at different levels of detail in the form of a tree-like structure. Which generalizes all the different process models in the respective subtree.

The FSM Miner/Petrify method proposed by Rubin et al. (2006) concentrates on extracting control flow at the various levels of event log abstraction. Basically it follows a two-step approach. Initially, a transition system constructed from the traces in an event log. Finally, Petri net is discovered by synthesizing the transition system. This method fails to handle noise. A method named Genet, which is very similar to FSM Miner/Petrify has been proposed by Carmona et al. (2010). It allows the production of a Petri net from a transition system, and suffers from the similar drawbacks of FSM Miner/Petrify.

Inductive miner (Leemans et al., 2013a) presented a process discovery framework B, which is capable of generating block structured process model which is sound and fit. It is further proved that B produces a set of sound, fitting models in finite amount of time. Extension of Inductive Miner - infrequent (IMi). An extension of the Inductive Miner (IM, called B$'$) is proposed by Leemans et al. (2013b), that filters infrequent behavior locally in each algorithmic step of IM by selecting an operator and a cut, splitting the log and base cases of the recursion.

D Other approach for control flow visualization

Splitpar algorithm proposed by Herbst and Karagiannis (2004) is part of the InWoLvE framework for process analysis. It extracts a structured process model by initially deriving stochastic activity graph out of event log. Splitpar is capable of detecting redundant activities. Workflow miner (Gaaloul et al., 2005), models the elementary dependencies in the event log and enables the discovery of structural workflow patterns in the form of intermediary graphical representation. Enhanced version of workflow miner proposed by Greco et al. (2006), which addresses important challenges (such as noise, duplicate tasks, and non-free choice) in control flow discovery . This method is also capable of discovering control flow from an unlabeled event stream (with the help of Expectation Maximization method).

Van Dongen and Van Der Aalst (2005) demonstrated the method of discovering a process model by aggregating group of related process models into a petri net. It was achieved by applying Event-driven Process Chains as an intermediate step to derive an executable process model.

ILP Miner (Van derWerf et al., 2009) uses the Integer Linear Programming to control flow discovery. This method, previously famous as Parikh language-based region miner, is built on the famous theory of regions and it permits for parallelization. It was shown to be free from the number of events in the event log, and the method can be much useful in real-life settings.

One of the notable exception for visualizing control flow process model is the Multiple Trace Alignment (MTA) (Bose and Van Der Aalst, 2010). Trace alignment is a way of arranging process traces to identify the areas of similarity that may be a consequence of *functional*, *structural* or *evolutionary* relationships between the activities of the process traces.

A representation named dotted chart (Song and Van Der Aalst, 2007) shows the process events in graphical way such that end user gets a bird's eye view of overall process trace execution scenario. Dotted chart shows the spread of events of an event log over time. The basic idea of the dotted chart is to plot dots according to the time. It has time and component as orthogonal dimensions. It offers large array of opportunities for process management and improvement.

3.4.2 Process modeling notations

Brief overview of most frequently used control-flow modeling notations in process mining is presented in this section. Table 3.2 lists the summary of information shown by each of the control flow modeling notations discussed in this section.

Summary of the various information shown by each of the modeling notation is listed in

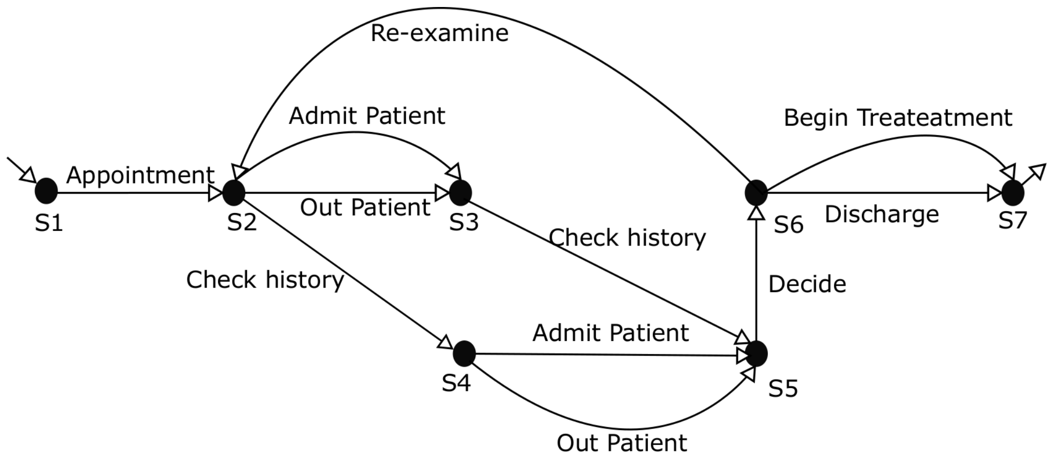


Figure 3.4: Hospital admission process in transition net notation.

A Transition systems

Transition system (Van Der Aalst et al., 2004) is a most rudimentary control flow modeling notation. It consists of states and transitions. It is represented using triplet

$TS = \{S, A, T\}$. Where S represents set of states,

- $A \subseteq \mathcal{A}$ is a set of activities and T is set of transitions ($T \subseteq S \times \subseteq A \subseteq S$).
- $S^s \subseteq S$ and $S^e \subseteq S$ denotes the set of initial and final states.

Transition systems are simple, but tend to fail when they are used for representing concurrent systems due to state explosion problem (Valmari, 1998). Hospital admission process modeled in transition system notation is given in fig. 3.4.

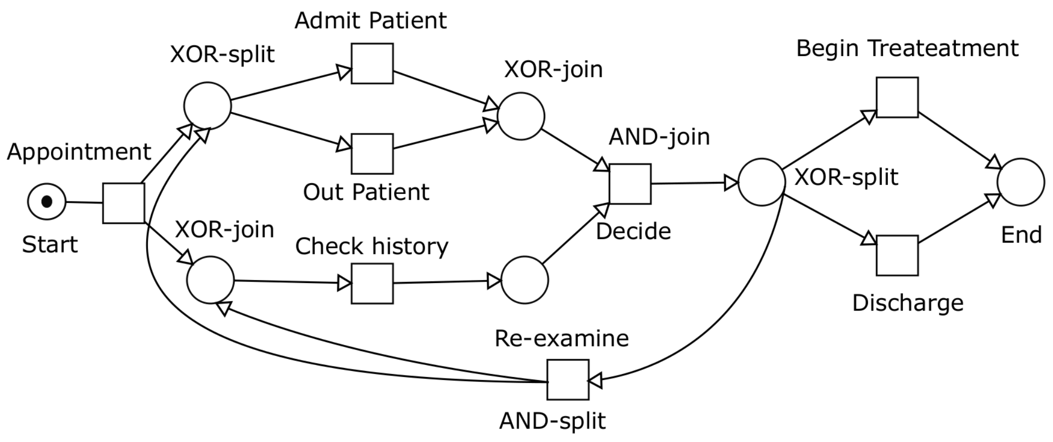


Figure 3.5: Hospital admission process in petri net notation.

B Petri net

α algorithm (Van Der Aalst et al., 2004) available in process mining can generate control flow of a process in the petri net notation. Petri net model of hospital admission process is given in fig. 3.5.

A petri net is a triplet $N = (P, T, F)$ where,

- P is a set of places.
- T is a set of transitions such that $P \cap T = \phi$.
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs called flow relations.

Petri net consists of places and transitions. The structure of petri net model is static, but it is controlled by firing rule. Distribution of tokens over network is referred as marking and it determines the state of petri net.

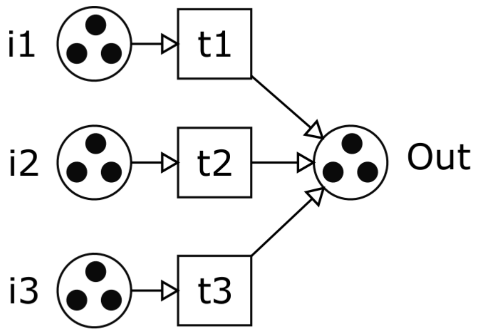


Figure 3.6: Hospital admission process in workflow net notation.

C Workflow nets

A workflow net is a extension of petri nets with a specific start and end place (Van Der Aalst, 1998). A workflow net is represented with attributes as $N = (P, T, F, A, L)$. (a) N is a workflow net if and only if P contains an input place i such that $\bullet i = \phi$, (b) P contains an output place o such that $o\bullet = \phi$, and (c) There should be directed path between any pair of nodes in N. Representation of hospital admission process using workflow nets is given in fig. 3.6.

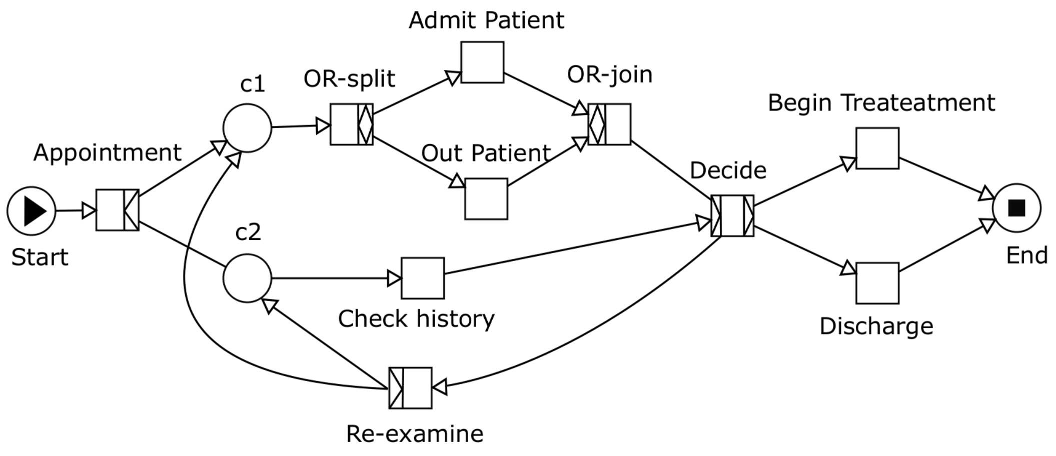


Figure 3.7: Hospital admission process in YAWL notation

D Yet Another Workflow Language (YAWL)

YAWL is a amalgamation of workflow system and modeling language. Development of YAWL was greatly driven by the workflow patterns initiative (Van Der Aalst et al., 2003). Based on a regular investigation of the notations utilized by existing control flow notations and workflow languages, a group of patterns was considered. These patterns consists of control flow, data, resource, change, exception patterns, etc.

The purpose of YAWL is to extend a easy assistance for many patterns while keeping the language simple. Activities in YAWL are called tasks. Tasks can be associated with different split/join types (for example: AND split/join, OR split/join, XOR split/join, etc.). Further, a task in YAWL can be composite or atomic. A composite task refers to another hierarchical sub-model. Fig. 3.7 gives the YAWL representation of hospital admission process.

Table 3.2: Control flow visualization notations and information depicted in them

	Set of information shown in a process model						
Method	*Activities*	*Transition*	*Order of activities*	*General Consensus*	*Information Score*	*Conserved and shared activities*	*Relative occurace*
Transition systems	✓	✓	✓	×	×	×	×
Petri net	✓	✓	✓	×	×	×	×
Workflow net	✓	✓	✓	×	×	×	×
Yet Another Workflow Language	✓	✓	✓	×	×	×	×
Business Process Modeling Notation	✓	✓	✓	×	×	×	×
Event Driven Process Chain	✓	✓	✓	×	×	×	×
Causal net	✓	✓	✓	×	×	×	×
Dotted chart	×	×	×	×	×	×	×
Trace alignment	✓	×	✓	✓	✓	×	×

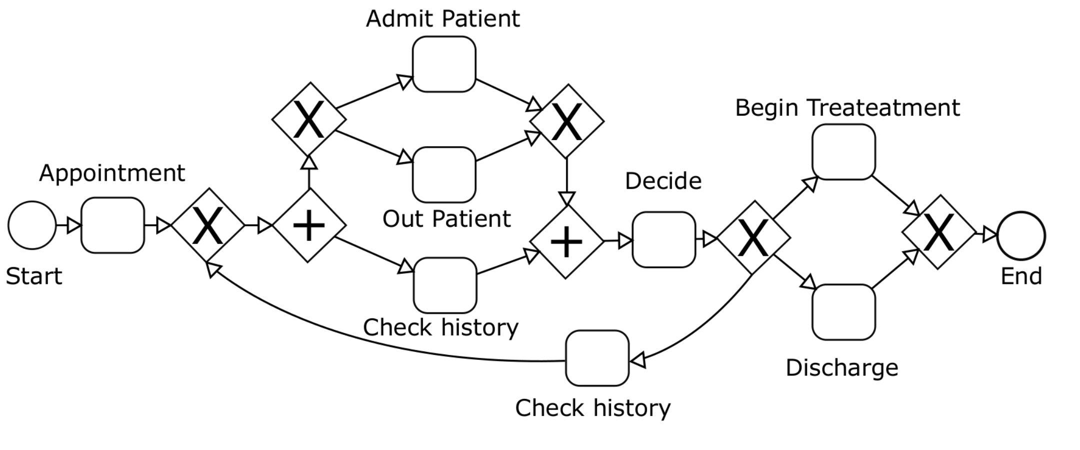


Figure 3.8: Hospital admission process in BPM notation.

E Business Process Modeling Notation (BPMN)

In recent times BPMN (White et al., 2004) has became widely used notation to model operational processes. It preserves the concept of tasks from YAWL. Unlike YAWL, in BPMN splits/joins are not associated with tasks but with separate gateways. BPMN model of hospital admission process is shown in fig. 3.8.

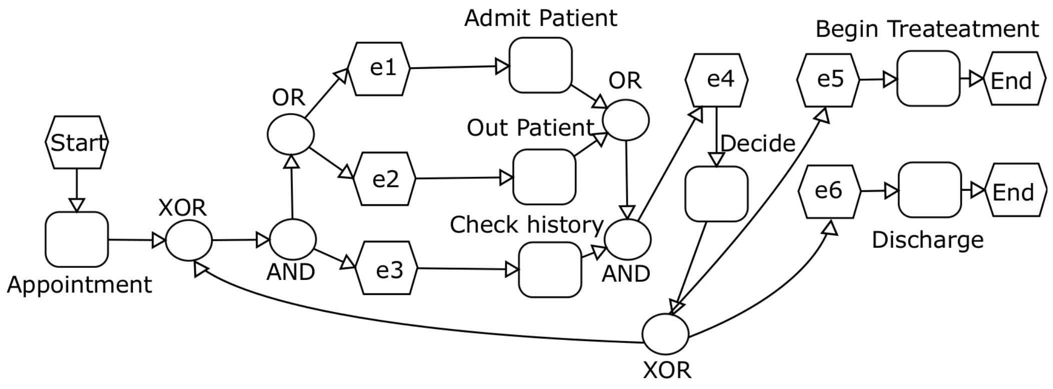


Figure 3.9: Hospital admission process in EPC notation.

F Event Driven Process Chains (EPC)

EPCs (Scheer et al., 2005) fundamentally offers a subset of feature from BPMN and YAWL with it's own graphical notations. Activities in EPCs are called as functions. Functions consists of one input and output arc. Connectors are used for modeling splits and joins of type OR, AND and XOR. EPC consists of three types of events namely start, intermediate and end. Functions and events need to alternate along any path, i.e., it is not allowed to connect any functions to functions and events to events.

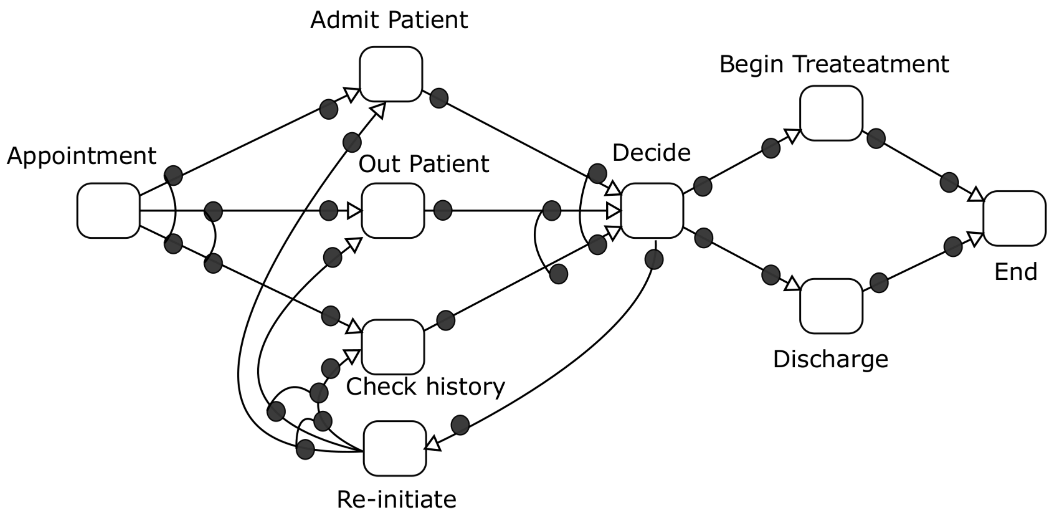


Figure 3.10: Hospital admission process in causal net notation.

G Causal nets

A causal net (Adriansyah et al., 2010) is a graph where arcs signify causal dependencies and nodes signify activities. Each activity has a set of possible input and output

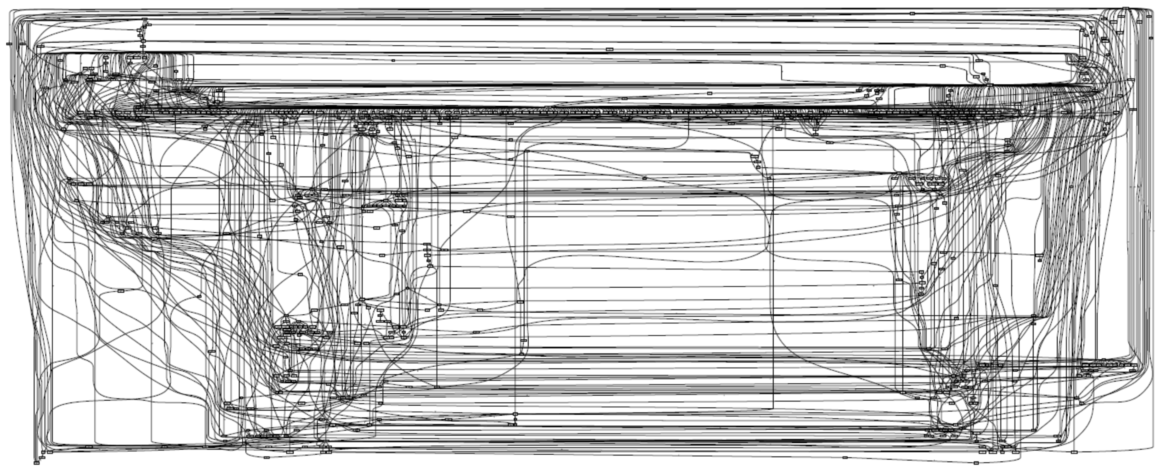


Figure 3.11: Spaghetti process related to diagnosis treatment of patients in dutch hospital.

bindings which guides routing logic in causal net. However, there are no places in the causal net; the routing logic is solely represented by the possible input and output bindings.

Causal nets are highly suitable for control flow related tasks of process mining. It is due to their declarative property and expressiveness without using all classes of extra model elements (places, conditions, events, gateways, etc.). Casual net representation of hospital admission process is shown in fig. 3.10.

3.5 Structured and Unstructured Process

In the initial stages of process mining research, synthetically produced event logs were used to develop and validate the proposed mining techniques. These methods will create a comprehensive depiction of the observed behavior (in the form of a control flow process model). While the results obtained by using synthetically generated event logs are convincing, but contradict when the same techniques are evaluated on real-life event logs. It is due to the fact that, most of the real life processes are not carried out in a strict and rigid workflow management systems. Inflexible characteristics of information systems compelled most of the business vendors to choose more flexible ad-hoc solutions. It is apparent that conducting a process within flexible operational environments will lead to highly diverse and unstructured behavior.

Case Handling (Van Der Aalst et al., 2005) and Adaptive workflow management systems (Han et al., 1998) permit operators to alter or define procedures in a flexible style which does not firmly outline an exact track of execution. However, the most common solutions for executing operational processes do not apply any well-defined behavior, yet merely extend the functionality for data and message sharing between operators and resources. Examples of these systems are Enterprise Resource Planning, plain text E-Mail, custom-built solutions, or Computer-Supported Cooperative Work systems. This flexibility in process execution has exposed a basic flaw in most of the initial process mining algorithms. When these algorithms are employed for analyzing the event logs of less-structured processes, the outcome is unstructured process models which are difficult to comprehend and use for operational support. The control flow models generated by mining less structured processes resemble to Spaghetti.

It is important to notice that these Spaghetti models are not incorrect. The processes themselves are really Spaghetti-like, i.e., the model is an accurate reflection of reality. An example of spaghetti process model related to diagnosis and treatment of $2,765$ patients in dutch hospital is shown in fig. 3.11. This process model was constructed based on an event log containing $114,592$ events over 619 activities executed by 266 different individuals.

Spaghetti models are insightful, but extracting useful results from it is difficult. The sheer amount of activities in Spaghetti model makes it difficult to focus on the significant parts. Total number of edges in the process model which forms the Spaghetti structure poses a challenge in understanding. Spaghetti structure reflects the heart of flexibility in process execution when users and resources are permitted to carry out anything in any order. It renders monitoring business activities fundamentally not an easy task. The problems lie neither with unstructured processes nor with process mining practices. Fairly, it is due to numerous implicit assumptions in process mining techniques, both at the level of processes and event logs. These assumptions are valid in controlled environments and fail to hold in less-structured and real-life settings.

On the other part of process complexity, relatively structured control-flow models are referred as Lasagna process. The executions through Lasagna processes are guided in controlled manner. Hence, the most of the available process mining techniques generate compelling results when they are employed for analyzing Lasagna processes. Complexity of process ranges from highly structured to unstructured and it is continuum. The process complexities sometimes referring to structured, semi-structured and unstructured refers to same continuum.

- In a *structured process* (such as Lasagna), all activities are repeatable and have a well defined input and output.
- In *semi structured processes* the information requirements of activities are known and it is possible to sketch the procedures followed. However some activities require human judgment and people can deviate depending on taste or the characteristics of the case being handled.
- In the context of *unstructured processes* (such as Spaghetti), it is impossible to define post and pre-conditions for activities. These processes are driven by qualitative information, rules-of-thumb, trail-and-error, intuition, and experience.

3.5.1 Spaghetti structured processes

Spaghetti processes and Lasagna processes are the complement of each other. Due to less structured nature of Spaghetti processes, only a subset of the available techniques in process mining can be practiced. For example, it is highly impossible to aim at operational support activities if there is huge variability. Still, process mining can support to understand and facilitate the process improvements by revealing important problems.

Researchers have came up with various solutions for simplifying and analyzing Spaghetti structured processes. The initial methods for analyzing Spaghetti process was proposed by Greco et al. (2006). They had proposed the iterative and hierarchical refinement of the process model, where, at each step, traces sharing similar behavior patterns are clustered together and equipped with a specialized schema. The resulting model is a disjunctive schema that explicitly takes care of variants of the process.

Fuzzy mining (Günther and Van Der Aalst, 2007) provides the method for adaptive process simplification. It uses the concept of a road-map as a metaphor to visualize the resulting models. Based on an analysis of the log, the importance of activities and relations among activities are taken into account. Activities and their relations can be clustered or removed depending on their role in the process. Certain aspects can be emphasized graphically just like a road-map emphasizes highways and large cities over dirt roads and small towns.

Bose and Van Der Aalst (2009a) has proposed a multiphase method for characterizing and identifying frequently observed control flow constructs (such as loops, activity sequences, and splits/joins) in the process and a means to form abstractions over identified patterns. Typically, abstracting common process constructs results in simplified process models.

Method of clustering process traces based on the contextual information is proposed by Bose and Van Der Aalst (2009c). This method partitions the traces into clusters such that the process models mined from those clusters show a high degree of fitness and that the models are more comprehensible.

Any operational processes are exciting from the perspective of process mining as they often permit for numerous enhancements. A structured process is often less exciting in this aspect; it is easy to employ process mining methods but there is no or less room for improving the existing process. Hence, Spaghetti processes are often interesting from a process management viewpoint. Converting Spaghetti to Lasagna processes is highly helpful for an Organization.

3.6 Summary of Literature Survey

This section briefs the significant issues noted from the literature survey. The methods and techniques solving the issues identified in this section are solved in the upcoming contents of this thesis.

- Concept drift refers to an online supervised learning scenario when the relation between the input data and the target model changes over time. Concept drift phenomenon is a significant issue in most of the data analysis discipline. Currently available techniques in process mining are poor at analyzing the processes with concept drift. There are very limited attempts to address this issue in process mining. The method proposed by Bose et al. (2014) suffers from the feature set dimensionality problem and the one suggested by Carmona and Gavalda (2012) fails to detect the multiple concept drifts in the process.

  There is a need for methods and techniques which are capable of handling it with respect to various perspectives. In this thesis chapter 4 address this issue by proposing the methods and techniques for detecting and localizing sudden concept drift in control flow perspective of the operational process.

- In the section 3.4, a various control flow discovery algorithms and modeling notations have been discussed. Discovery algorithms aim at discovering the process semantics, and modeling notations enable to express the same processes in various forms (depending on the context of use). The main underlying information depicted in all of the process notations and discovery techniques is activities and connections between them. Frequency, consensus, information score, conserved and shared activities are not shown in traditional methods.

  The notation named process logo presented in chapter 5 offers a technique to visualize the control flow of process with additional and important information in a single compact graphic.

- Process model becomes complex and cannot be used if there are a sheer number of activities and transitions in it. One such category of the process model is named as Spaghetti. Spaghetti models are important from the aspect of process mining; they are normally observed in many real life processes where there is no strict enforcement of information system.

  The need is to simplify the structure of Spaghetti process models to extract the simplified process model. Which should be representing the significant parts of the process (in terms of activity and transition). This simplified model is used for operational support, communicating with the stake holders and for identifying the set of possible or impossible paths of executions. A method has been proposed the chapter 6 to address this issue.

- Process models play vial role in organizations for design, redesign and implementation of information systems. To achieve this, formal informal models can be used. But, ability to determine the frequently executed path in the underlying process model benefit the stakeholders in many ways for understanding, communicating, discussion, optimization, etc. A method has been proposed in chapter 7 to find out the frequently executed paths in information system.

Based on the previous discussions, we present the problem statement and the objectives of current research work in the upcoming sections.

3.7 Problem Statement

Design and development of improved process mining techniques for handling *concept drift*, *analyzing spaghetti processes*, and informative *envisioning* of control flow process models.

3.8 Research Objectives

1. Propose a framework for detecting and localizing sudden concept drift in control-flow perspective of an operational process.
2. Propose a control-flow notation for depicting general consensus among traces, order of prevalence of activities, relative occurrences, information score and set of shared/conserved activities.
3. Distilling Lasagna structured control-flow from Spaghetti structured control-flow.
4. Identify and predict the possible and frequent path of execution in Spaghetti structured process.

Chapter 4

Detecting and localizing control flow concept drift in process mining

It is apparent that the financial growth of a business organization is greatly centered on competence to respond to changes in its functioning environment. To cope with competition, it is highly essential for the organizations to update the processes to reduce the cost and increase productivity. Furthermore, customers expect enterprises to be adaptable and adjust to altering conditions.

New laws (such as WABO act (Thomas, 2015) and Sarbanes-Oxley act (Welytok, 2006)), *excessive differences in resource and request*, *periodic effects*, *catastrophes*, *time limit violation* (Van Der Aalst et al., 2007b), and so on, are also compelling organizations to change their processes. Therefore, the topic of flexibility is well researched in the domain of BPM and workflow management (WFM). But still, a majority of process mining algorithms considers the operational process as a static entity.

Today's process discovery techniques are able to extract meaningful process models from event logs not containing any explicit process information. When discovering a process model from event logs, in process mining, it is assumed that the process at the beginning of the recorded period is the same as the process at the end of the recorded period. Obviously, this is often not the case due to the phenomenon known as *concept drift* (Bose et al., 2011). While cases are being handled, the process itself may be changing. This chapter presents an approach to analyze such second-order dynamics.

Concept drift is a situation when the process changes during the period of execu-

tion/analysis. Due to concept drift, the structure of the process at the cessation differs from its commencement, and it makes the end-result of analysis obsolete. Ability to handle concept drift efficiently is the most critical aspect of ensuring correctness and validity of the end-results produced by methods in process mining.

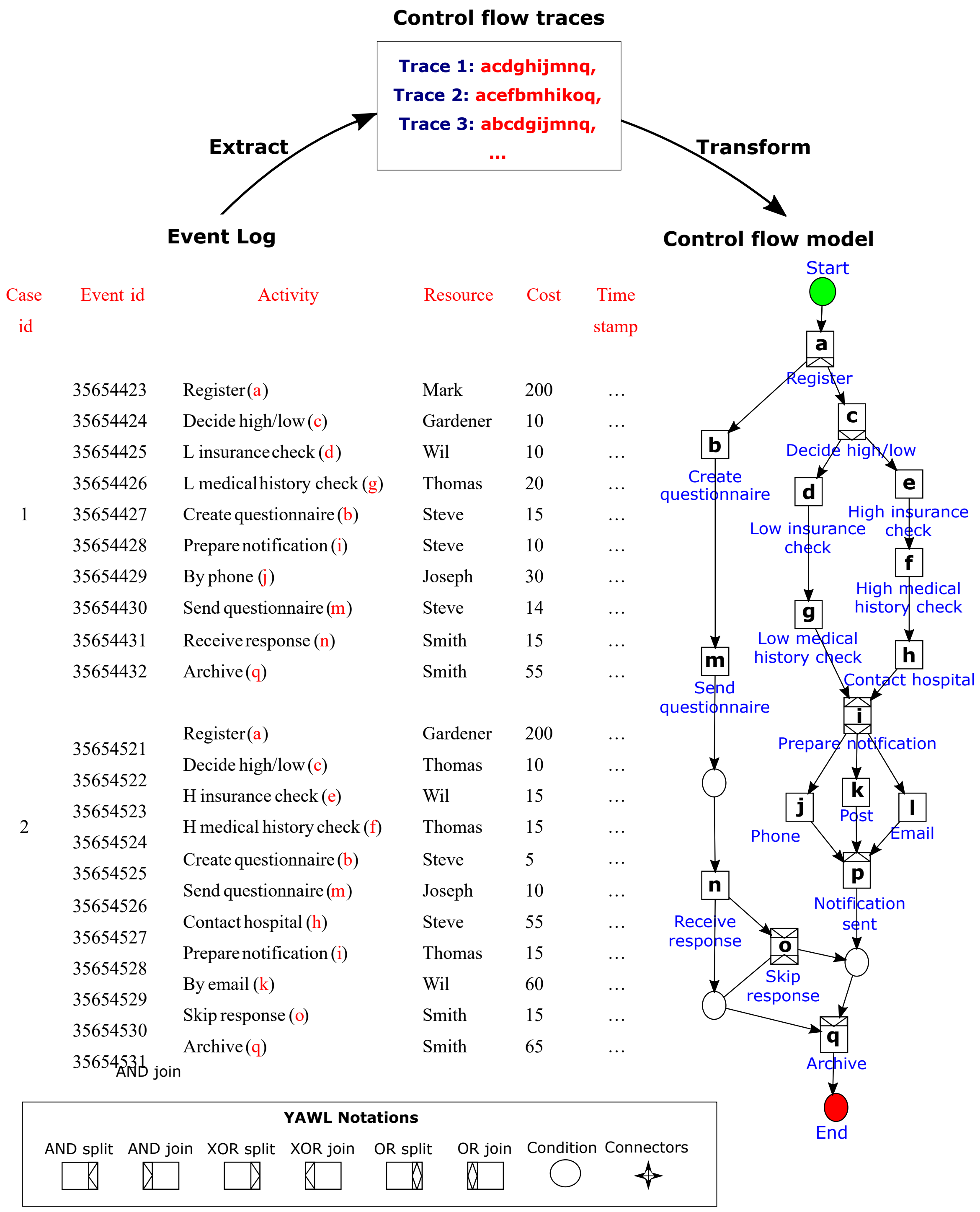
In process mining, *ProM* is an extensible and open-source framework that provides a variety of techniques in the form of plug-ins. There are more than 1200 plug-ins available in *ProM* and none of them are capable of addressing concept drift completely (Van Der Aalst et al., 2012; Van Der Aalst and Weijters, 2004). Process mining manifesto (Van Der Aalst et al., 2012) is an article released by IEEE CIS Task Force on Process Mining. That enlists eleven open research directions. Handling concept drift is one among the most pressing eleven challenges. To this point in time, a little work is done on this highly significant problem (Bose et al., 2014; Carmona and Gavalda, 2012).

We propose the *offline* technique for *detecting* and *localizing* concept drift in the *control-flow* perspective of operational process. Features for concept drift detection and localization are derived by processing consensus traces generated by MTA. Subsequent populations of feature values are compared by applying an appropriate hypothesis tests to handle concept drift.

Upcoming contents are organized as follows, section 4.1 describes the event log related to insurance claim process, section 4.2 presents the basic problems to be addressed to handle concept drift and also illustrates concept drift with a small example. MTA has been presented in section 4.3. Feature values used for handling concept drift are described in 4.4. Section 4.6 describes the experimental setup. Results of experimental study is given in section 4.6. Comparison results related to existing and proposed method for concept drift handling is presented in section 4.7.

4.1 Insurance Claim Event Log

The method for concept drift detection and localization proposed in this chapter are validated on the real life event log of insurance claim process. The excerpt of the event log and initial control flow connections of insurance claim process is shown in the figure 4.1.



Trace 1: acdghijmnq,
Trace 2: acefbmhikoq,
Trace 3: abcdgijmnq,
...

Case id	Event id	Activity	Resource	Cost	Time stamp
1	35654423	Register (a)	Mark	200	...
	35654424	Decide high/low (c)	Gardener	10	...
	35654425	L insurance check (d)	Wil	10	...
	35654426	L medical history check (g)	Thomas	20	...
	35654427	Create questionnaire (b)	Steve	15	...
	35654428	Prepare notification (i)	Steve	10	...
	35654429	By phone (j)	Joseph	30	...
	35654430	Send questionnaire (m)	Steve	14	...
	35654431	Receive response (n)	Smith	15	...
	35654432	Archive (q)	Smith	55	...
2	35654521	Register (a)	Gardener	200	...
	35654522	Decide high/low (c)	Thomas	10	...
	35654523	H insurance check (e)	Wil	15	...
	35654524	H medical history check (f)	Thomas	15	...
	35654525	Create questionnaire (b)	Steve	5	...
	35654526	Send questionnaire (m)	Joseph	10	...
	35654527	Contact hospital (h)	Steve	55	...
	35654528	Prepare notification (i)	Thomas	15	...
	35654529	By email (k)	Wil	60	...
	35654530	Skip response (o)	Smith	15	...
	35654531	Archive (q)	Smith	65	...

Figure 4.1: Initial control-flow model of insurance claim process (inspired from Bose et al. (2014)).

According to the initial control flow structure of insurance claim process shown in fig. 4.1, upon registration (a) of any travel insurance claim, a list of questions will be prepared (n) and sent to the claimant (o). Simultaneously, based on the amount, the claim is classified as high or low (b). If the claim amount is high, three steps of verification are carried out serially (check medical history (d), check insurance (g), and contact hospital (i)). Else (c), two steps of verification (insurance check (e) and medical history check (f)) are carried out serially. Insurance claim is rejected or accepted based on the verification outcome. Notification of acceptance or rejection is prepared (j) and sent to the claimant. Result of the process is communicated through phone (m) or post (k) or email (l). Answers from the claimant for questionnaire is considered (q) or ignored (p) depending on the situation. Finally, the case is closed and put in archive (r).

4.2 Concept Drift

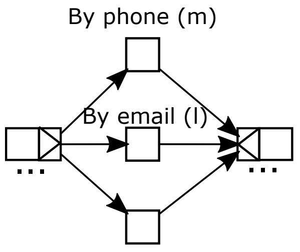
This section discusses the sub-problems involved in solving the problem of concept drift and illustrates the concept drift in control flow perspective by modifying the causal relationships between a subset of activities related to insurance claim process.

4.2.1 An example illustrating concept drift in control-flow perspective

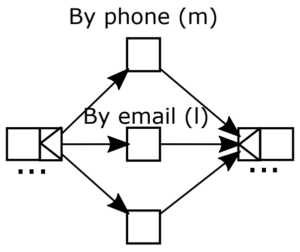
Concept drift in control-flow perspective is the result of change in causal relationships between the activities. Same is illustrated by modifying the causal relationships between activities *by post* (k), *by phone* (l) and *by call* (m) of *insurance claim* process model shown in fig. 4.1. For representational simplicity, further sections of this chapter uses YAWL notations for representing process models.

Initially, activities k, l, and m are arranged parallel between XOR split/join (shown in fig. 4.2a), with the occurrence of successive concept drifts cd_1 (fig. 4.2b), cd_2 (fig. 4.2c), and cd_3 (fig. 4.2d); causal relationships between the activities change as follows.

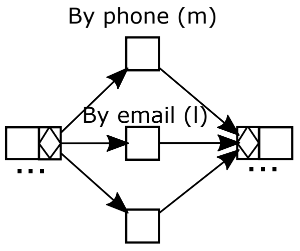
- Concept drift cd_1 arranges the activities k, l, and m parallelly between AND split/join (shown in fig 4.2b). This control-flow arrangement forces all three activities to execute simultaneously (irrespective of their order of occurrence).
- Concept drift cd_2 arranges the activities k, l, and m parallelly between OR split/join (shown in fig 4.2c). This control-flow arrangement permit the execution of any combination of these three activities .



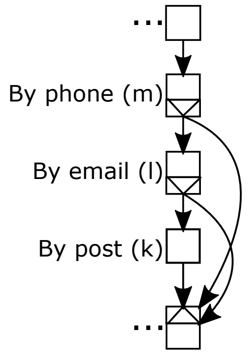
(a) Before any concept drift (at time t_0). Activities `k, l` and `m` are arranged parallel between XOR split/join.



(b) After cd_1 (at time t_1). Activities `k, l` and `m` are arranged parallel between AND split/join.



(c) After cd_2 (at time t_2). Activities `k, l` and `m` are arranged parallel between OR split/join.



(d) After cd_3 (at time t_3). Activities `k, l` and `m` are arranged in knockout fashion.

Figure 4.2: Change in causal relationships between activities.

- Concept drift cd_3 arranges the activities `k, l,` and `m` in knock-out fashion (shown in fig 4.2d). In this arrangement, process execution can skip any of the intermediate activities to reach the final one.

Process excerpts given in fig. 4.2 illustrates the causal relationship between the same set of activities after successive concept drifts cd_1, cd_2, and cd_3.

Table 4.1: Control-flow traces before cd_1, after cd_1, cd_2, and cd_3.

Time	**Concept drift**	**Set of possible traces**
t_0	Before cd_1	$\mathcal{A}_1$ = {`k` ,`l`,`m`}
t_1	After cd_1	$\mathcal{A}_2$ = {klm, kml, mkl, mlk, lmk, lkm}
t_2	After cd_2	$\mathcal{A}_3$ = {`k`, l, m, kl, lk, lm, ml, km, mk,,klm, kml, mkl, mlk, lmk, lkm}
t_3	After cd_3	$\mathcal{A}_4$ = {`k`, kl, klm}

Table 4.2: Possible number of traces in the process and sub-process level.

	Number of possible traces at the sub-process level			**Number of possible traces at the process level**		
Concept drift	$\mid \mathcal{A}_{cd_i} \mid$	$\mid \mathcal{A}_{cd_i}/\mathcal{A}_{cd_{i-1}} \mid$	$\mid \mathcal{A}_{cd_i} \cap \mathcal{A}_{cd_{i-1}} \mid$	$\mid \mathcal{P}_{cd_i} \mid$	$\mid \mathcal{P}_{cd_i}/\mathcal{P}_{cd_{i-1}} \mid$	$\mid \mathcal{P}_{cd_i} \cap \mathcal{P}_{cd_{i-1}} \mid$
Before concept drift	3	-	-	12	-	-
After cd_1 at t_1	6	6	$\emptyset$[1]	24	24	$\emptyset$
After cd_2 at t_2	15	9	6	60	36	24
After cd_3 at t_3	3	$\emptyset$	3	12	$\emptyset$	12

The set of possible control-flow traces on activities `k,l` and `m` before cd_1, after cd_1, cd_2 and cd_3 are given in table 4.1. In the table 4.2,

- $\mathcal{A}_{cd_i}$ and $\mathcal{P}_{cd_i}$ represent the set possible traces at sub-process and process level.
- $\mathcal{A}_{cd_i}/\mathcal{A}_{cd_{i-1}}$ and $\mathcal{P}_{cd_i}/\mathcal{P}_{cd_{i-1}}$ is the set difference.
- $\mathcal{A}_{cd_i} \cap \mathcal{A}_{cd_{i-1}}$ and $\mathcal{P}_{cd_i} \cap \mathcal{P}_{cd_{i-1}}$ represent set intersection.
- $\mid \mathcal{A}_{cd_i}/\mathcal{A}_{cd_{i-1}} \mid$ and $\mid \mathcal{P}_{cd_i}/\mathcal{P}_{cd_{i-1}} \mid$ are the number of newly observable traces after every concept drift cd_i.
- $|\mathcal{A}_{cd_i} \cap \mathcal{A}_{cd_{i-1}}|$ and $|\mathcal{P}_{cd_i} \cap \mathcal{P}_{cd_{i-1}}|$ gives the number of common traces that can be observed in the sub-process and process level prior to and after every cd_i.

Concept drift can either increase or decrease the number of possible traces. For example,

- After the concept drift cd_1, the process is capable of exhibiting double the number of traces possible before cd_1. We can observe six traces at the sub-process level after cd_1, all of them are entirely new(intersection between the set of traces before and after cd_1 evaluates to disjoint set ($\emptyset$)).
- Set of traces observed after cd_3 are the subset of set of traces possible before cd_3 (i.e., after cd_2).

The causal relationship between the activities change to satisfy the immediate demands and tend to exhibit new behaviors. It is wrong to consider an operational process as a static entity while analyzing an event log.

[1]Empty set is represented by $\emptyset$

The focus of this chapter is to propose a robust *offline* method to *detect* and *localize sudden/abrupt concept drift* in the *control-flow perspective* of the process. Addressing concept drifts in the perspectives except *control-flow*, and change patterns other than *sudden* are out of the scope of this research work. The framework followed for handling the sudden control-flow concept drift is illustrated in fig. 4.3.

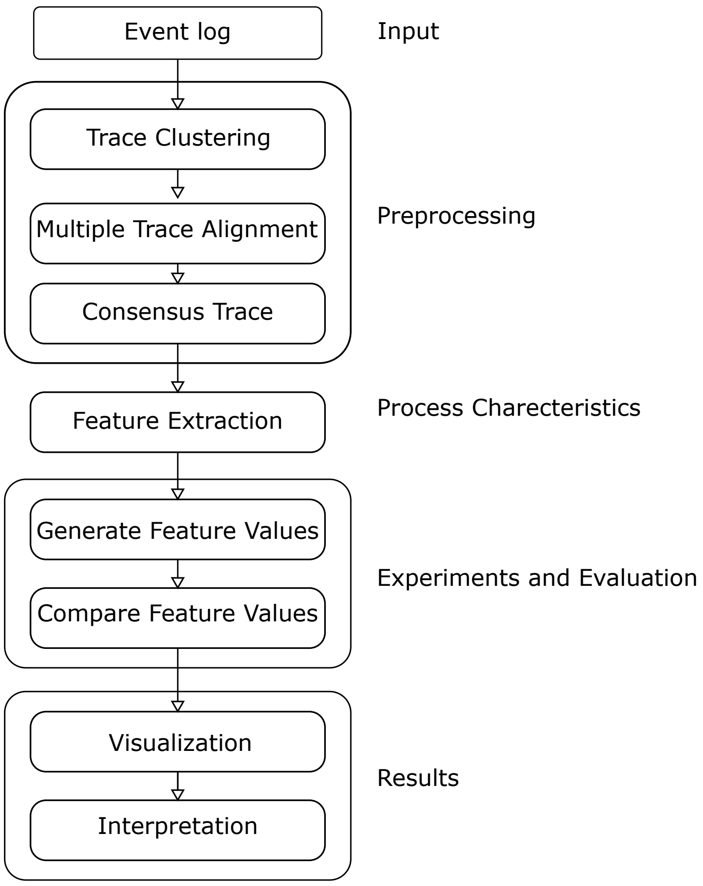


Figure 4.3: Framework for detecting and localizing concept drift.

4.3 Trace Alignment to Detect Concept Drift in Process

Trace alignment is a way of arranging process traces to identify the areas of similarity that may be a consequence of *functional*, *structural* or *evolutionary* relationships between the activities of the process traces. The idea of trace alignment was proposed by Bose and Van Der Aalst (2010) based on the concept of *biological sequence alignment* (Waterman, 1995).

Trace alignment is used as a *preprocessing stage* applied on traces. Based on the similarities, traces in the event log are clustered, and each cluster is aligned to generate a consensus trace, which is used as the input component for extracting features to capture the control-flow specific characteristics. This section uses following notations for discussing the concepts related to feature set.

- Σ is the set of activities and $|\Sigma|$ denotes the number of activities.
- Σ^+ is the set of all non-empty finite traces over Σ and $T \in \Sigma^+$ is a trace.
- An event log $\mathcal{L}$ corresponds to a set of traces from Σ^+.
- A trace of length n is denoted as T^n, i.e., $T^n \in \Sigma^n$ (where, Σ^n is the set of all n-length traces).
- Activities of T^n are denoted as $T(1)T(2)T(3)...T(n)$, where $T(k)$ represents k^{th} activity in trace.

4.3.1 Multiple Trace Alignment (MTA)

MTA is a method of aligning more than two traces simultaneously. Trace alignment over a set of traces can be represented as $\mathcal{A} = \{a_{i,j}\}(1 \leq n, 1 \leq j \leq m)$ over $\Sigma^{'} = \Sigma \cup \{-\}$ where, $-$ denotes the gap in the alignment. Trace alignment over a set of traces $\mathbb{T} = \{T_1, T_2, T_3..., T_n\}$ is *defined* as mapping of a set of traces $\bar{\mathbb{T}} = \{\bar{T}_1, \bar{T}_2, \bar{T}_2, ..., \bar{T}_n, \}$ where, each $\bar{T}_i \in (\Sigma \cup -)^+$ for $1 \leq i \leq n$ and

- $|\bar{T}_1| = |\bar{T}_2| = ... = |\bar{T}_n| = m.$
- $\bar{T}_i$ by removing all "$-$" gap symbols is equal to T_i.
- $\nexists k$, $1 \leq k \leq m$ such that, $\forall_{1 \leq i \leq n}$. $\bar{T}_i(k) = -$

Where, m is the length and n is the number of traces in the alignment.

The span of the alignment m, satisfies the relation $l_{max} \leq m \leq l_{sum}$. Where, l_{max} is the maximum length of the traces in $\mathbb{T}$ and l_{sum} is the sum of lengths of all traces in $\mathbb{T}$. It is important to note that there can be many possible alignments for a specified set of traces. The number of possible alignments for two traces of length l is given by equation 4.1.

$$l \approx (1 + \sqrt{2})^{(2l+1)} l^{(\frac{-1}{2})} \tag{4.1}$$

Traces are aligned after a series of *substitution*, *insertion*, and *deletion* (indel) operations. A score-function assigns the value for each substitution and indel operations. The scores associated with substitution and indel operations are used to calculate the optimal alignment.

- The substitution score is a function $S : \Sigma \times \Sigma \rightarrow \Re$ where, $S(\texttt{a,b})$ denotes the score for substitution of activity $\texttt{a}$ with activity $\texttt{b}$, $\forall \texttt{a, b} \in \Sigma$.
- IndelRightGivenLeft score is a function $\mathcal{I}_l : \Sigma \cup \{-\} \times \Sigma \cup \{-\} \rightarrow \Re$ where, $\mathcal{I}_l(\texttt{a, b})$ indicates the score for inserting activity $\texttt{a}$ given that the left activity is $\texttt{b}$, $\forall \texttt{a,b} \in \Sigma$.

The best alignment is calculated using the aggregate sum of the number of substitutions, insertion, and deletion operations are applied to align a set of given traces. An alignment with the maximum score is considered as optimal. We use sum-of-pairs (Bose and Van Der Aalst, 2010) method for automatically deriving the optimal substitution and indel scores from the event log.

If $\bar{T_1}$ and $\bar{T_2}$ are the traces after alignment and if the length of the alignment is m, then the score of the alignment can be calculated using the equation 4.2. The value of e_i in equation 4.2 is calculated using equation 4.3.

$$Score(\bar{T_1}, \bar{T_2}) = \sum_{i=1}^{m} e_i \tag{4.2}$$

$$e_i = \begin{cases} S(\texttt{a,b}) & \text{if } \overline{T}_1(i) = \texttt{a} \text{ and } \overline{T}_2(i) = \texttt{b} \\ \mathcal{I}_l(\texttt{a,b}) & \begin{cases} \text{if } \overline{T}_1(i) = \texttt{a}, \overline{T}_1(i-1) = \texttt{b} \text{ and } \overline{T}_2(i) = \,\breve{}\, \text{ or} \\ \text{if } \overline{T}_1(i) = \,\breve{}\,, \overline{T}_2(i) = \texttt{a} \text{ and } \overline{T}_2(i-1) = \texttt{b} \text{ or} \end{cases} \end{cases} \tag{4.3}$$

For example, fig. 4.4 represents three of many possible alignments between the traces $T_1 = \texttt{abcdabe}$ and T_2=$\texttt{acddcbc}$ (according to formula $(1+\sqrt{2})^{(2l+1)} l^{(\frac{-1}{2})}$, 2×10^5 alignments are possible between T_1 and T_2). By considering the scoring function $S(\texttt{a,b}) = 1$ if $\texttt{a=b}$ else -1 and $\mathcal{I}_l(\texttt{a,b}) = -1$ $\forall \texttt{a, b} \in \Sigma$, the scores of alignments shown in figures 4.4a - 4.4c evaluates to -2, 0, and -4 respectively, and a alignment shown in fig. 4.4b, with the maximum score is considered as optimal.

Alighnamet score	-1	-1	+1	+1	-1	-1	+1	-1
$\overline{T}_1$	a	b	c	d	a	-	b	e
$\overline{T}_2$	-	a	c	d	d	c	b	c
Total score	-1	2	-1	0	-1	-2	-1	**-2**

(a) First possible alignment

Alighnamet score	+1	-1	+1	-1	+1	-1	+1	-1
$\overline{T}_1$	a	b	c	-	d	a	b	e
$\overline{T}_2$	a	-	c	d	d	c	b	c
Total score	+1	0	+1	0	+1	0	+1	**0**

(b) Second possible alignment

Alighnamet score	+1	-1	-1	+1	-1	-1	-1	-1
$\overline{T}_1$	a	b	c	d	-	a	b	e
$\overline{T}_2$	a	c	d	d	c	b	c	-
Total score	+1	0	-1	0	-1	-2	-3	**-4**

(c) Third possible alignment

Figure 4.4: Alignments showing three of the 2×10^5 possibilities between two traces.

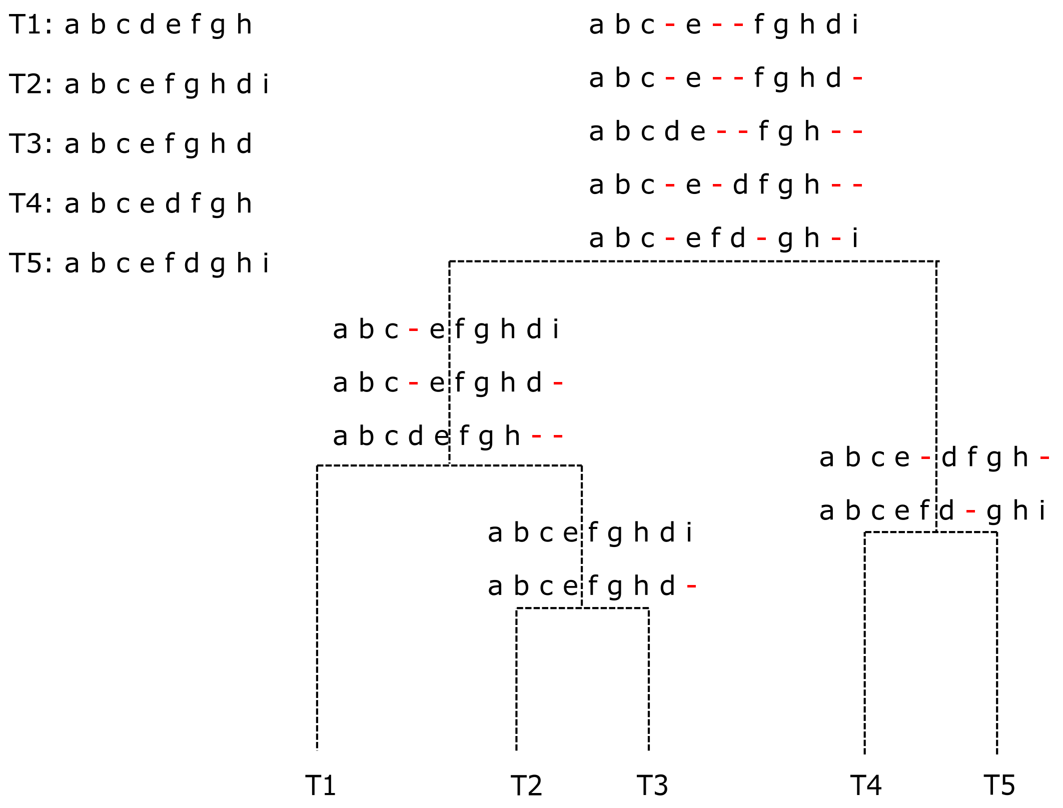


Figure 4.5: Example of progressive alignment technique with the help of AHC.

In the preprocessing stage of producing alignment, duplicate traces from the event log are filtered out, and distinct traces are clustered using the *Agglomerative Hierarchical Clustering algorithm* (AHC) (Day and Edelsbrunner, 1984). A *dynamic* *programming* (Bertsekas, 1995) method known as *Needleman and Wunsch algorithm* (Needleman and Wunsch, 1970) is used to find optimal alignment between a pair of traces.

Finding the globally optimal alignment between multiple process traces can be done by repeated application of the Needleman and Wunsch algorithm on every pair of traces incrementally with the help of the progressive alignment technique. Further, the optimal alignment corresponding to every stage of the output of AHC is built. Fig. 4.5 shows the example of aligning five traces with the help of AHC and progressive alignment technique. The result is aligned cluster of unique traces. Further, AHC can be cut at any levels to form required number of aligned trace clusters.

Figure 4.6 illustrates the result of aligning initial 1000 traces of *insurance claim* process (fig. 4.1). Duplicate traces are filtered out (selected 18 distinct traces out of 1000 traces), and distinct traces are clustered based on their similarity. Trace alignment technique is applied on each cluster of traces to generate the optimal alignment. For example, result of grouping the same set of traces into one , two (cluster 1: 6 traces and cluster 2: 12 traces), and three (cluster 1: 6 traces, cluster 2: 6 traces, cluster 3: 6 traces) clusters and aligning each cluster as shown in fig. 4.6. Each alignment is capable of showing following set of information as a single compact graphic,

- Each row in the alignment represents a transformed trace, and each column represents an activity.
- The left panel shows the case ids (as in the log), and case ids with a gray background specify traces that have matching duplicates.
- Each MTA shows split/joins (if any) and order of arrangement of activities.
- The bottom panel shows the information score metric for each column and consensus trace corresponding to all traces of alignment.

It is observed that, dividing a given set of traces with increasing number of clusters (1, 2, 3,..) results in reduced uncertainty (entropy) and improved information score (Shannon, 2001). This is due to the groping of coherent set traces in a same cluster. Features related for concept drift detection and localizations are extracted out of MTAs resulting from aligning traces of each cluster.

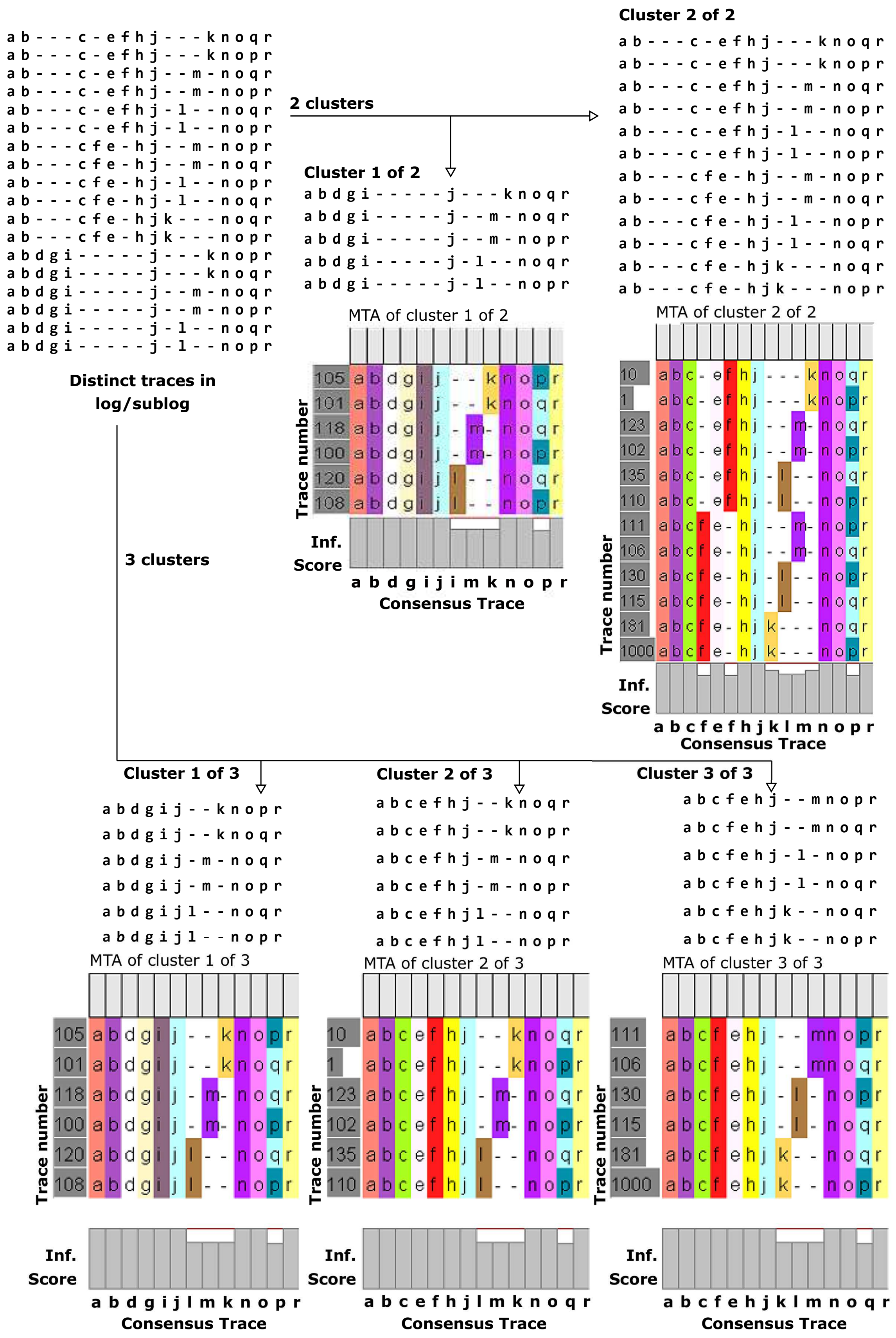


Figure 4.6: Result of MTA on 1000 traces consisting of 18 distinct activities.

4.3.2 Consensus trace

The consensus trace captures the major activity in each column of the trace alignment and can be considered as the backbone trace representing the control flow of the process. Consensus trace is capable of compactly representing the following information.

- Consensus of the traces.
- Order of prevalence of the activities at every position.
- Relative occurrences of every activity with respect to other activities.
- List of conserved and shared activities in the aligned set of traces.

`abdgicfefhjklmknoqr` (cluster 1 of 1), `abdgijlmknopr` (cluster 1 of 2, cluster 1 of 3), `abcfefhjklmnopr` (cluster 2 of 2), `abdgijlmknopr` (cluster 1 of 3), `abcefhjlmknoqr` (cluster 2 of 3) and `abcfehjklmno` (cluster 3 of 3) are the consensus traces corresponding to multiple trace alignments shown in fig. 4.6.

Corresponding to the trace alignment given in fig. 4.6 (two clusters). The relative frequency of the activities in consensus traces 1 and 2 of second alignment is given in table 4.3. Columns with relative frequency 1.0 indicate well-conserved patterns. For example, in consensus trace `abdgijlmknopr`, activities `a, b, d, g, i, j, n, o` and `r` are *conserved* and `m, k` and `p` are *shared.*

In the experimentation part, the complete event log is transformed into a set of consensus traces, and features are extracted by processing the consensus traces. Feature values are systematically analyzed to uncover the concept drift.

4.4 Features for Detecting and Localizing Concept Drift

Information score, consensus activity relationships, activity relationship entropy, and *window count* features are extracted from the consensus trace. We believe that there is a characteristic difference in the manifestation of the feature values before and after the process change, the change in feature values are more pronounced at the boundaries of the process change. Features that assist in the concept drift detection and localization are discussed in detailed as follows.

Table 4.3: Attributes and properties of consensus traces.

Total number of traces: 1000, 2 clusters.

Cluster: 1 of 2

Cluster size: 481 traces

Number of distinct traces: 6

Maximum information score: 4 bits

Alignment width: 13 activities

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Activities	a	b	d	g	i	j	l	m	k	n	o	p	r
Relative frequeny	1.0	1.0	1.0	1.0	1.0	1.0	0.32	0.32	0.34	1.0	1.0	0.49	1.0
Information score	3.97	3.97	3.97	3.97	3.97	3.97	1.0	1.0	1.0	3.97	3.97	1.47	3.97

Cluster 2 of 2

Cluster size: 519 traces

Number of distinct traces: 12

Maximum information score: 4.2 bits

Alignment width: 16 activities

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**		**16**
Activities	a	b	c	f	e	f	h	j	k	l	m	k	n	o	q	p	r
Relative frequeny	1.0	1.0	1.0	0.5	1.0	0.49	1.0	1.0	0.16	0.36	0.34	0.12	1.0	1.0	0.47	0.52	1.0
Information score	4.06	4.06	4.06	1.55	4.06	1.51	4.06	4.06	0.57	1.13	1.07	0.44	4.06	4.06	1.44	1.61	4.06

4.4.1 Information score

Information score (R_i) (Shannon, 2001) shows how well the activities are conserved at each position (column) in the alignment (it is measured in *bits*). It is calculated for every activity appearing in the column of the alignment. Higher the count of any particular activity in the column, higher will be the information score associated with that activity.

Activity with higher information score appears in consensus trace, which represents the predominant activity in that column. Information score decreases as the number of distinct activities in that column increases. The Information score of an activity at position i is given by equation 4.4.

$$R_i = \log_2(|\Sigma|) - (H_i + e_n) \tag{4.4}$$

Where, H_i (calculated using equation 4.5) is the uncertainty (sometimes called the Shannon entropy) of position i in the consensus trace.

$$H_i = -\sum f_{\mathsf{a},i} \times \log_2 f_{\mathsf{a},i} \tag{4.5}$$

Here, $f_{\mathsf{a},i}$ is the relative frequency of the activity at position i and e_n is the small-sample correction for an alignment of n activities. The information score of activity a in column i is given by equation 4.6.

$$height = f_{\mathsf{a},i} \times R_i \tag{4.6}$$

The approximation for the small-sample correction, e_n, is given by equation 4.7.

$$e_n = \frac{1}{\ln 2} \times \frac{s-1}{2n} \tag{4.7}$$

Where, s is the number of activities in the process and n is the number of traces of the process. Table 4.3 shows the information score of the consensus traces resulting from aligning the traces of insurance claim process shown in fig. 4.1.

4.4.2 Consensus activity relationships

The relationships between activities in a consensus trace can be expressed using *follows* and *precedes* notations. For any given pair of activities in Σ, one can find out whether they *sometimes*, *never* or *always follow/precede* each other or not in the consensus trace. The activity relationships are sufficient to discover most of the changes related to control-flow perspective.

For example, the set $\mathcal{L} = \{$ `abdgijlmknopr`, `abcfefhjklmnoqr`$\}$ represents the set of consensus traces of second alignment shown in fig. 4.6. Consensus activity relationships such as `b` always follows `a`, `j` sometimes follows `i`, and `e` never follows `d` always hold on set $\mathcal{L}$. Consensus activity relationship is used to derive *relation count, relation entropy*, and *window count* features. These features will be used during the concept drift detection.

A Relationship count

Relationship count is a function ($f_{rc} : \Sigma \rightarrow \mathbb{N}_0^3$) can be defined over *follows* or *precedes* relationship. For any activity, `a` $\in \Sigma$, f_{rc} is a vector of three integer values $\langle count_{always}$, $count_{sometimes}$, $count_{never}\rangle$. Where, $count_{always}$, $count_{sometimes}$, and $count_{never}$ are the number of activities in Σ that *always*, *sometimes*, and *never* follow `a` in $\mathcal{L}$.

For example, the relationship count of activity `o` on a set of consensus traces $\mathcal{L}$ is $\langle 1, 2, 14\rangle$. Activity `r` always *follow* `o`, activities `q and p` sometimes *follow* `o`, and all other activities except `p, q` and `r` never follow `o`.

B Relation entropy

Relations entropy is a function ($f_{re} : \Sigma \rightarrow \mathbb{R}^+$) can be defined over follows and precedes relationship of all activities in Σ. f_{rc} of any activity `a` $\in \Sigma$ with respect to a relationship count can be defined as,

$$f_{re} = p_a \log \frac{1}{p_a} + p_s \log \frac{1}{p_s} + p_n \log \frac{1}{p_n} \tag{4.8}$$

Where, $p_a = \frac{count_{always}}{|\Sigma|}$, $p_s = \frac{count_{sometimes}}{|\Sigma|}$ and $p_n = \frac{count_{never}}{|\Sigma|}$.

With respect to activity `o` in $\mathcal{L}$, $p_a = 0.05$, $p_s = 0.11$, and $p_n = 0.82$. The value of the relation entropy is $f_{re} = 0.24$.

C Window count

Window count is the function ($f_{wc} : \Sigma \times \Sigma \rightarrow \mathbb{N}_0$) defined over set of activity pairs based on *follows/precedes* activity relationships. For a given process instance p and a window length of size l, let S_l be the set of all sub-traces $p(i, i+l-1)$, such that $p(i)$ = `a` and $\exists\ j$ such that $i < j < i+l$ and $p(j)$ = `b`. The window count of the relation `b` follows `a` is defined as the number of traces of length l in which `b` follows `a`. In other words, $f_{wc}($`a,b`$) = |S_l|$.

For example, applying the *window count* function (f_{wc}) with window size of 4, on activities `a` and `b` in the consensus traces `abcdefgh`, `acdefgh`, `acdefb`, and `acdbefg` results in f_{wc} =1, 0, 0, and 1.

4.5 Experimental Setup

The goal of detecting and localizing concept drift is to detect the regions of changes and identify the nature of changes in the process log. There should be a typical dissimilarity in the manifestation of the feature values before and after the concept drifts with the difference being highly noticeable at the point of concept drift.

Initially, process log is split into sub logs containing n cases each. Traces in each sub log are clustered by applying AHC. MTA is preformed on each cluster and consensus traces are obtained. Which results in the transformation of event log into set of consensus traces. Further, duplicate consensus traces are removed. Feature extraction method discussed in section 4.4 are applied on the set of consensus traces to generate a population of feature values.

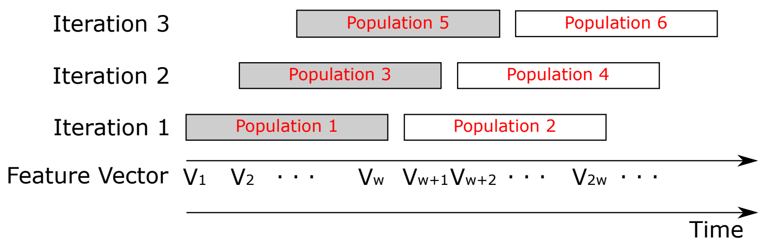


Figure 4.7: Feature value selection by windowing procedure.

Successive populations of feature values are selected for comparison with the help of windowing instance selection strategy shown in fig. 4.7. During every iteration of comparison, *population 1* ($\langle V_1, V_2 ... V_w \rangle$) and *population 2* ($\langle V_{w+1} ... V_{2w} \rangle$) are evaluated to discover any discrepancy. If m is the length of the vector and w is the window size, the possible number of population pairs are $1 - 2w + m$.

We use the method of *statistical hypothesis testing* (Sheskin, 2007) to compare and evaluate the successive populations of feature values. Hypothesis testing is used to determine the probability that a given statement is true based on the evidence available in the data. The standard process of hypothesis testing comprises of four steps.

- *First*, formulating *null* ($\mathcal{H}_0$) and *alternative hypothesis* ($\mathcal{H}_1$).
- *Second*, recognizing a test statistic that can be used to evaluate the correctness of the null hypothesis.
- *Third*, calculate the p value(lesser the p value, the stronger the evidence in contradiction of $\mathcal{H}_0$).
- *Finally*, compare the p value to α. If $p \leq \alpha$, then the null hypothesis is invalidated, and the alternative hypothesis is considered valid.

In our experiments, $\mathcal{H}_0$ and $\mathcal{H}_1$ are defined as follows,

- $\mathcal{H}_0$: There is no significance difference between the successive populations of feature values (i.e., $p > \alpha$).
- $\mathcal{H}_1$: There is a significant difference between the successive populations of feature values (i.e., $p \leq \alpha$).

Successive populations of feature values are compared with the application of *two-sample univariate* (tests dealing with scalar data), *multivariate* (tests dealing with vector data elements), and *non-parametric* (tests that does not follow any distribution) hypothesis test procedures.

We employed the *Siegel Tukey* and *Chi-square Test for $r \times c$ Tables* (Sheskin, 2007) for hypothesis evaluation. The *Siegel Tukey* test checks for the hypothesis, *"Do two independent samples represent two populations with different variances?"* and the *Chi-square Test* for the $r \times c$ tables evaluates the hypothesis, *"In the underlying* *population(s) represented by the sample(s) in a contingency table, are the observed cell frequencies different from the expected frequencies?".*

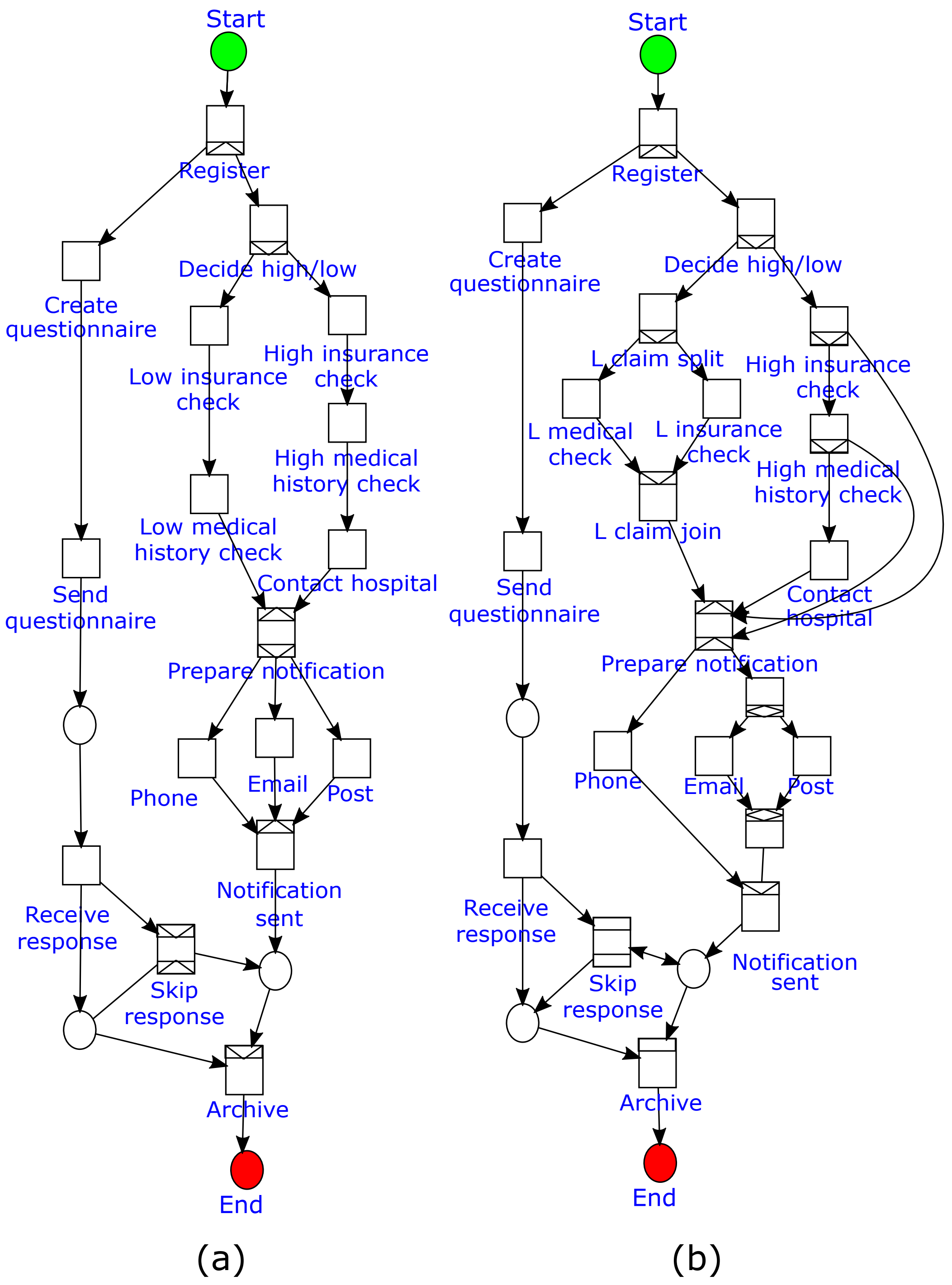
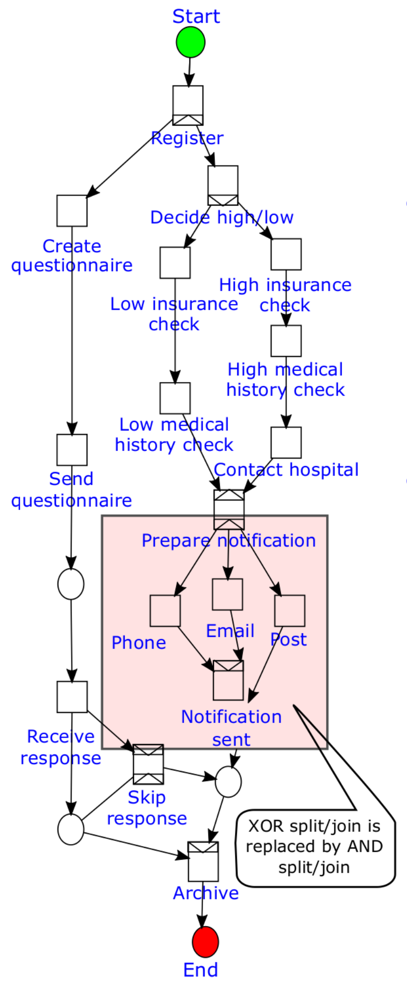


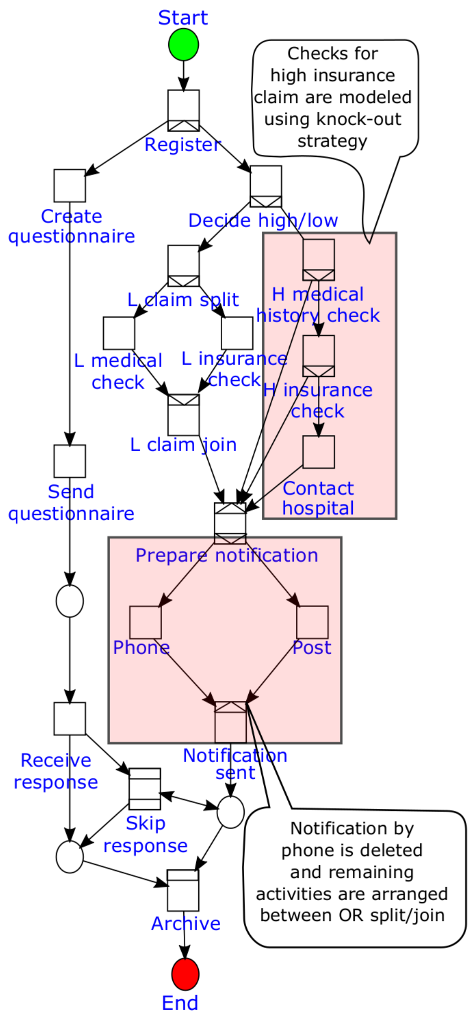
Figure 4.8: (a) Initial control-flow structure of insurance claim process (model m_0). (b) Final control-flow of insurance claim process after cd_5 (model m_5).

Table 4.4: Specific changes in the process leading to concept drift.

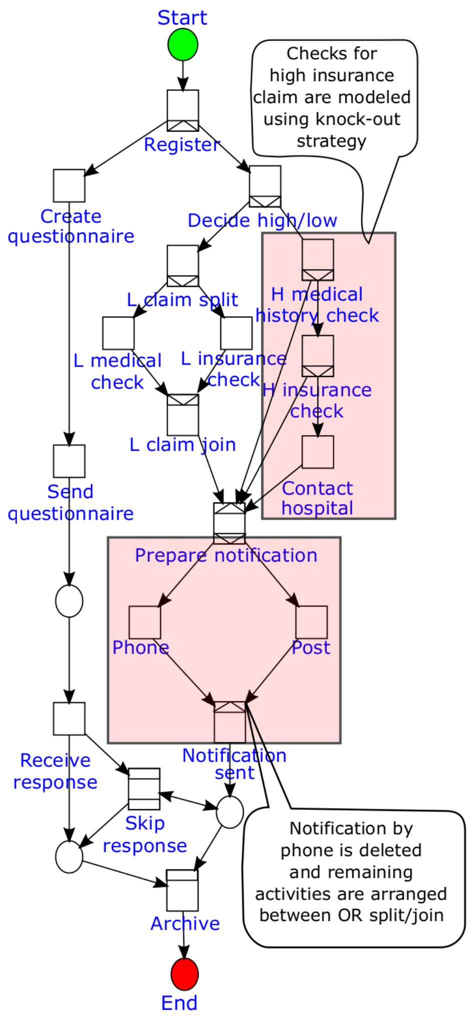
Concept drift	**Initial model**	**Changes**	**Transformed model**
$\boldsymbol{cd}_1$	m_0	• Activities *by post, by email* and *by phone* are arranged between AND split/join from XOR split/join. • This change will enable to send the notification to claimant through all the modes of communication, irrespective of their order.	m_1
$\boldsymbol{cd}_2$	m_1	• Serial relationship between activities related to *low* and *high* insurance claim is converted to parallel. • This change reduces the time required for verification. • Sub-tasks within high/low claim can be executed simultaneously. • Activities *by post, by email* and *by phone* are arranged between XOR split/join. • This change restricts insurance claim agency to send the acceptance/rejection of claim request through a single mode.	m_2
$\boldsymbol{cd}_3$	m_2	• Activities related to *high claim split* are arranged in *knock-out* fashion. • A request, however, can be rejected if anyone of the intermediate steps in high/low claim fail. In such cases, carrying out all the tasks related to high/low claims go waste. To rectify this, tasks related to checks are executed in knock-out (Van Der Aalst, 2001) fashion. In this arrangement, the next check will be done only if previous check results are positive, else the verification process will be terminated and the rejection notification will be sent to the claimant. • Activity *by phone* is deleted and remaining activities are arranged between OR split/join	m_3
$\boldsymbol{cd}_4$	m_3	• Causal relationship between the activities *by phone, by post* and *by email* are changed.	m_4
$\boldsymbol{cd}_5$	m_4	• Causal relationship between activities *by phone, by email,*and *by post* are changed. • This change makes notification through phone is mandatory.	m_5



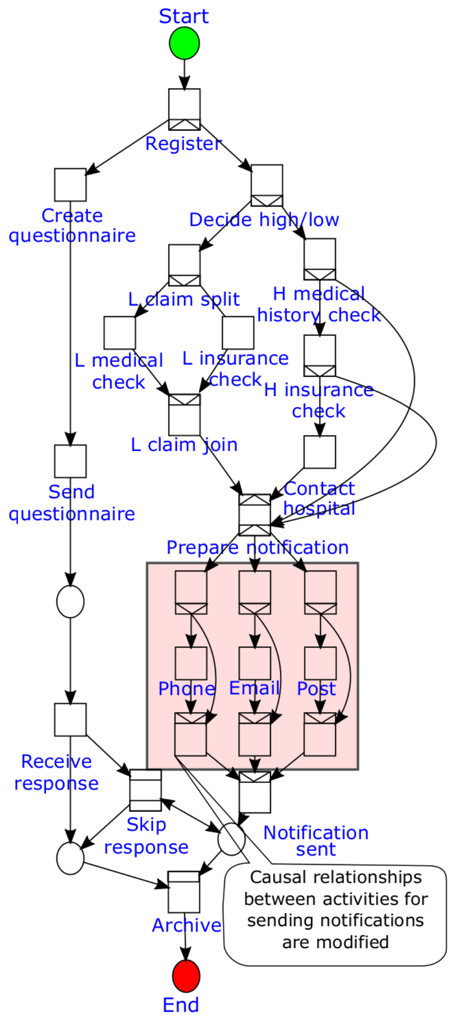
(a) Model m$_1$ (after cd_1).



(b) Model m$_2$ (after cd_2).



(c) Model m$_3$ (after cd_3).



(d) Model m$_4$ (after cd_4).

Figure 4.9: Control flow after successive concept drifts cd_1 to cd_4 (model m_1 to m_4).

4.6 Results and Discussion

The method for detecting and localizing the concept drift are evaluated on the event log of "insurance claim process". The event log consists of 48, 000 cases and 3, 84, 000 events over 20 distinct activities. The original event log (.xes file) used for the experimental study is modified to incorporate concept drift and regenerated using Colored Petri-Net tools (CPN tools)(Jensen and Kristensen, 2009) and CPNXES library(Michael, 2011).

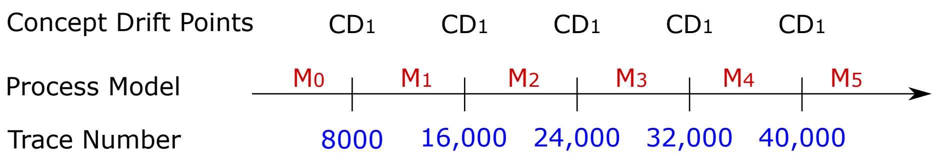


Figure 4.10: Concept drift locations.

The control-flow perspective of the insurance claim process shown in fig. 4.8 (initial control-flow) undergoes a series of sudden concept drifts. Process variants after successive concept drifts are shown in figs. 4.9a - 4.9d (intermediate variants) and 4.11 (final variant). Parts of the process highlighted in dotted rectangle represent the regions of change compared to it's previous variant. In other words, these are the parts of the process affected by concept drift. Set of changes that transformed and replaced a process model with its different variant are listed in the table 4.4. The concept drift locations and the process models during successive changes are shown in fig. 4.10. Changes are induced in *insurance claim* process log resulting in series of process variants. Previous process model is replaced by it's new variant for every 8000 traces.

MTAs corresponding to variants of insurance claim processes are shown in figures 4.12 - 4.17. Red-colored dashed rectangles on the alignments indicate control-flow concept drifts. MTAs demonstrates the nature of control-flow relationships between activities (serial, branch, parallel, etc.). Consensus trace corresponding to each MTA is given at the bottom of every alignment (highlighted in black color).

Fig. 4.12 shows MTA before any concept drift in the process (corresponding to process model given in fig. 4.8). MTA shown in the fig. 4.12 clearly shows Presence of XOR split at activity `b`, which allows either low(`c`,`d`) or high (`e`,`f` and `g`)

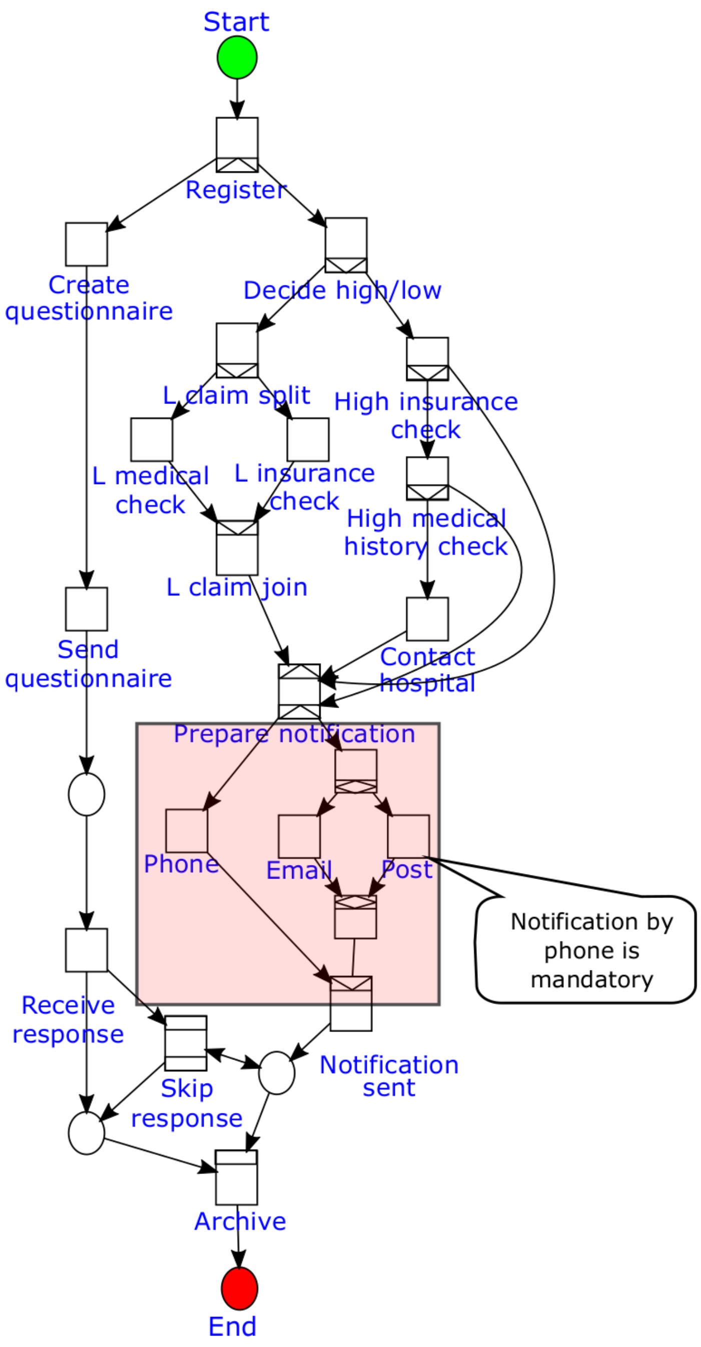


Figure 4.11: Final control-flow of insurance claim process after cd_5 (model m_5).

insurance verification. Similarly, presence of XOR split at activity `h` allows sending of notification through exactly one mode (either `i`, `j` or k). Fig. 4.13 shows MTA after the first concept drift cd_1 (corresponding to variant in fig. 4.9a). It clearly shows the presence of AND split at activity `h` which allows the usage of all mode of notification regardless of execution order.

Fig. 4.14 shows MTA after the second concept drift cd_2 (of process variant in fig. 4.9b). Fig. 4.15 shows MTA after the third concept drift cd_3 (of process variant in

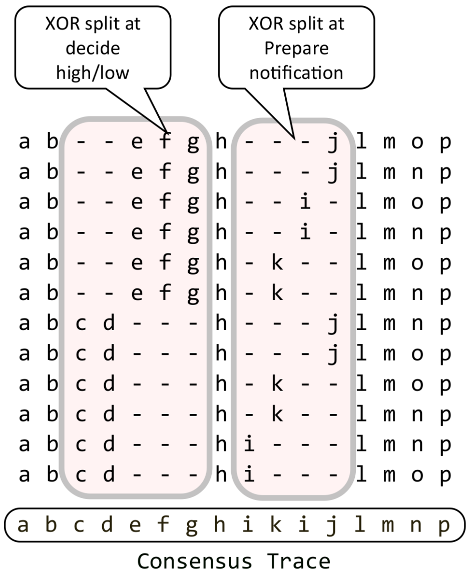

XOR split at decide high/low
XOR split at Prepare notification
a b - - e f g h - - - j l m o p
a b - - e f g h - - - j l m n p
a b - - e f g h - - i - l m o p
a b - - e f g h - - i - l m n p
a b - - e f g h - k - - l m o p
a b - - e f g h - k - - l m n p
a b c d - - - h - - - j l m n p
a b c d - - - h - - - j l m o p
a b c d - - - h - k - - l m o p
a b c d - - - h - k - - l m n p
a b c d - - - h i - - - l m n p
a b c d - - - h i - - - l m o p
a b c d e f g h i k i j l m n p
Consensus Trace

Figure 4.12: MTA of insurance claim process corresponding to model m_0.

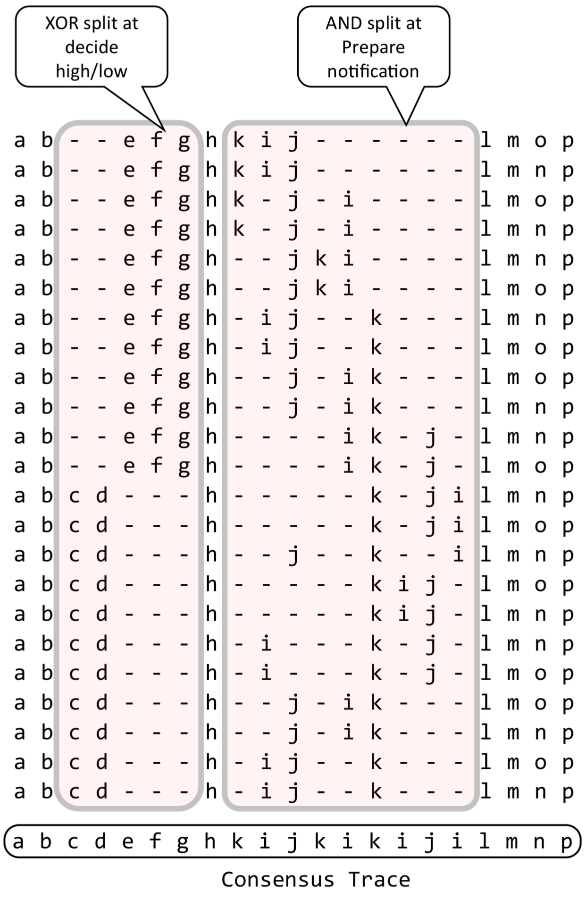

XOR split at decide high/low
AND split at Prepare notification
a b - - e f g h k i j - - - - - - l m o p
a b - - e f g h k i j - - - - - - l m n p
a b - - e f g h k - j - i - - - - l m o p
a b - - e f g h k - j - i - - - - l m n p
a b - - e f g h - - j k i - - - - l m n p
a b - - e f g h - - j k i - - - - l m o p
a b - - e f g h - i j - - k - - - l m n p
a b - - e f g h - i j - - k - - - l m o p
a b - - e f g h - - j - i k - - - l m o p
a b - - e f g h - - j - i k - - - l m n p
a b - - e f g h - - - - i k - j - l m n p
a b - - e f g h - - - - i k - j - l m o p
a b c d - - - h - - - - - k - j i l m n p
a b c d - - - h - - - - - k - j i l m o p
a b c d - - - h - - j - - k - - i l m n p
a b c d - - - h - - - - - k i j - l m o p
a b c d - - - h - - - - - k i j - l m n p
a b c d - - - h - i - - - k - j - l m n p
a b c d - - - h - i - - - k - j - l m o p
a b c d - - - h - - j - i k - - - l m o p
a b c d - - - h - - j - i k - - - l m n p
a b c d - - - h - i j - - k - - - l m o p
a b c d - - - h - i j - - k - - - l m n p
a b c d e f g h k i j k i k i j i l m n p
Consensus Trace

Figure 4.13: MTA after cd_1 (corresponding to model m_1).

fig. 4.9c). Fig. 4.16 shows MTA after the forth concept drift cd_4 (of process variant in fig. 4.9d). Single clustered MTA (of insurance process in fig. 4.9d) results in 150 distinct traces, such a huge alignment is impossible to show. Hence 150 traces are

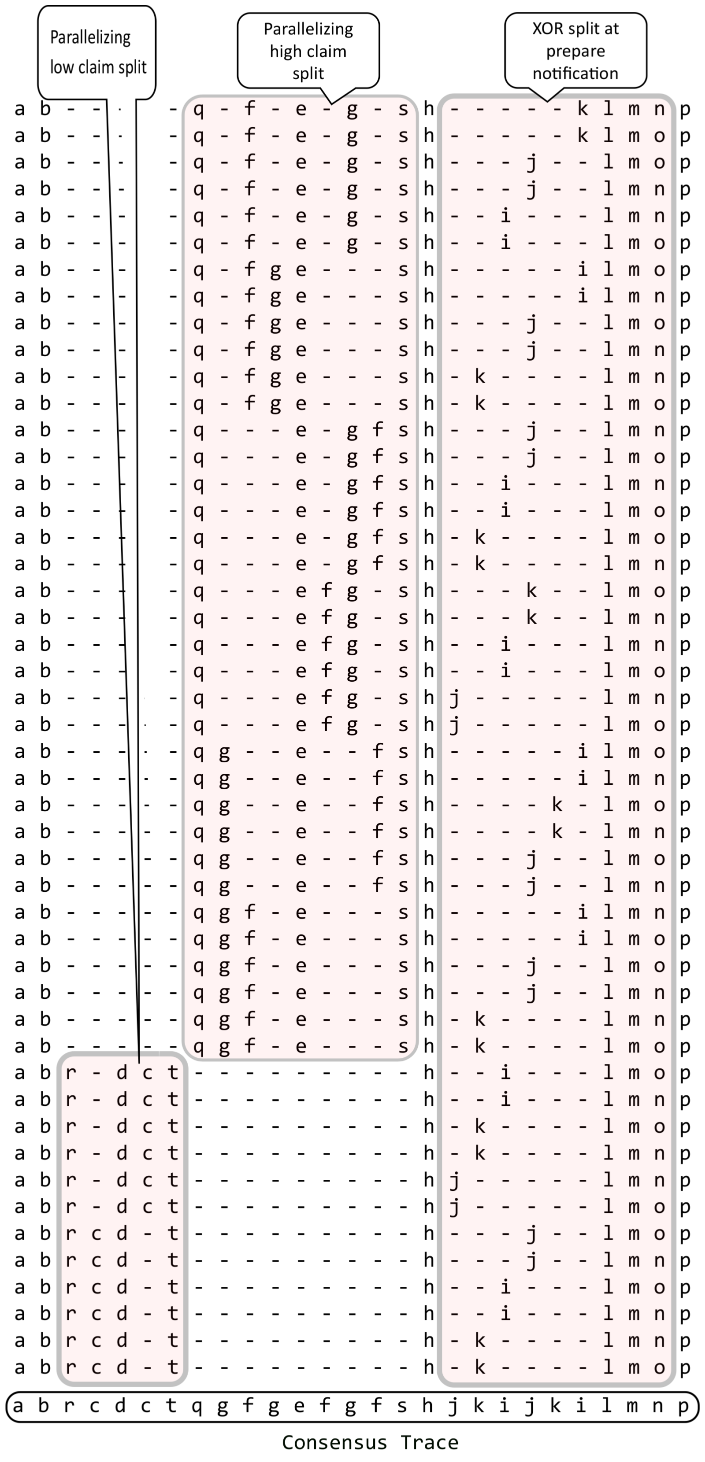
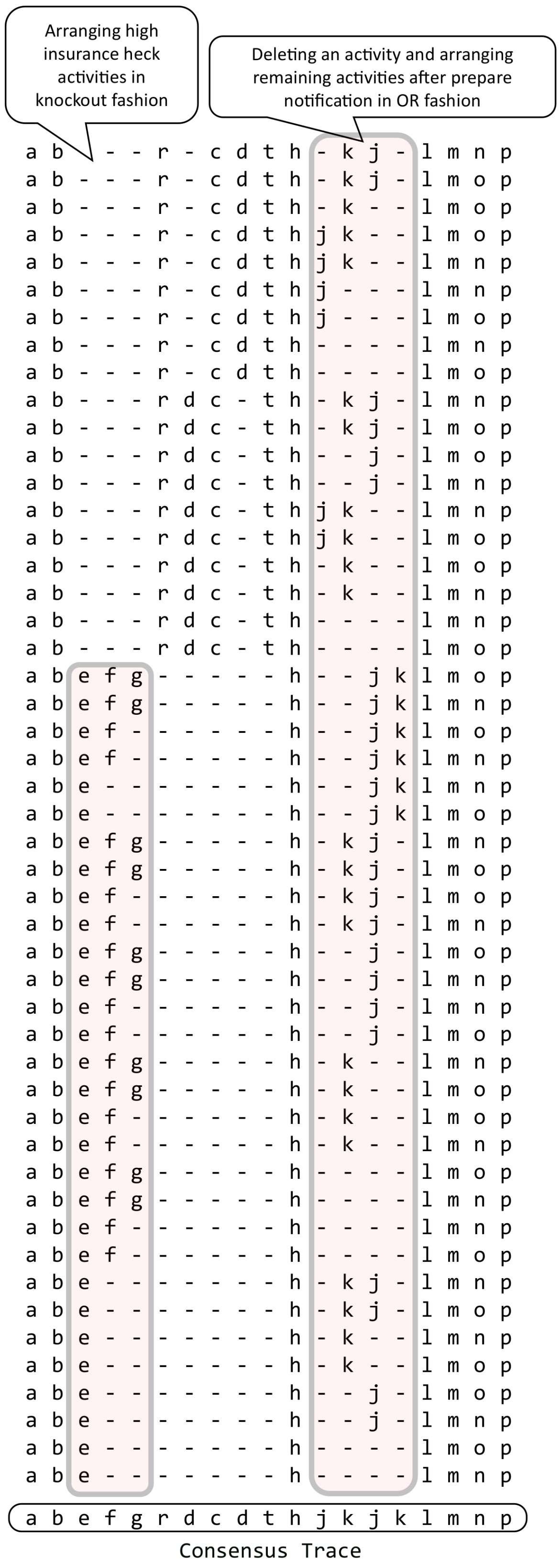

Parallelizing low claim split
Parallelizing high claim split
XOR split at prepare notification
a b r c d c t q g f g e f g f s h j k i j k i l m n p
Consensus Trace

Figure 4.14: MTA after cd_2 (corresponding to model m_2).

divided to form 3 clusters and MTA is generated. The MTA shown here is generated using 32 distinct traces of 2nd cluster. Fig. 4.17 shows MTA after the final concept drift cd_5 (of process variant in fig. 4.11).

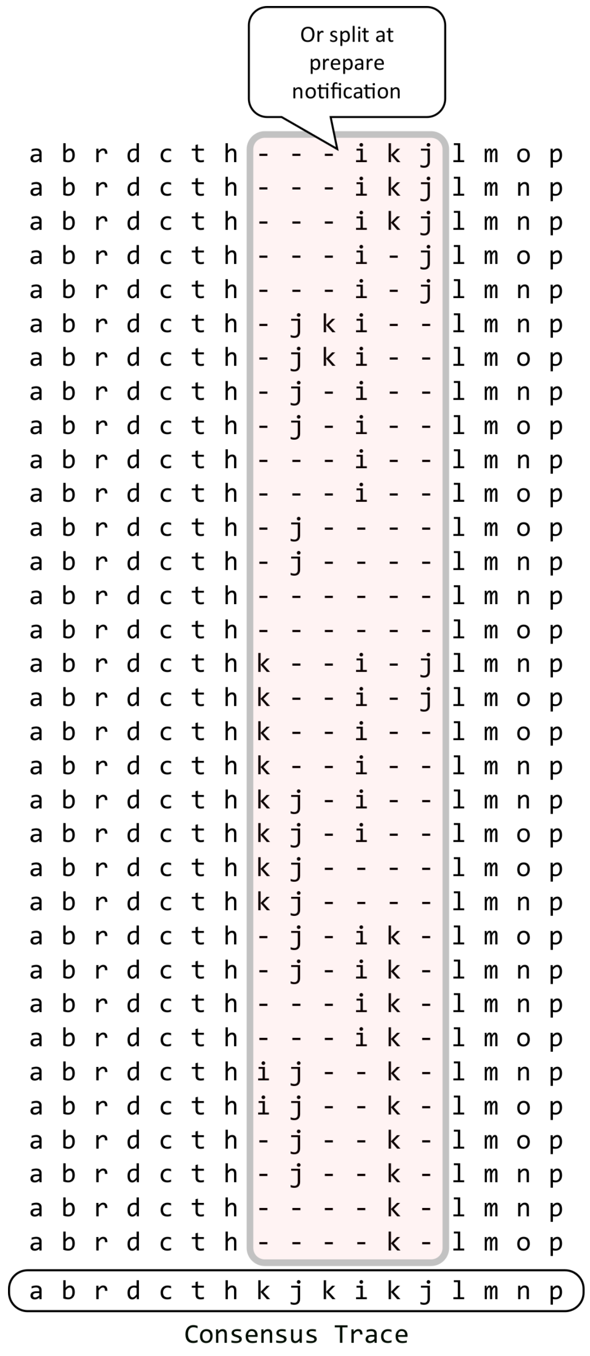
The 48,000 cases of the insurance claim process are split into sub log of 100 cases each, resulting in 480 sub logs. Redundant cases are filtered out from each sub log,


Arranging high insurance heck activities in knockout fashion
Deleting an activity and arranging remaining activities after prepare notification in OR fashion
Consensus Trace

```
a b - - - r - c d t h - k j - l m n p
a b - - - r - c d t h - k j - l m o p
a b - - - r - c d t h - k - - l m o p
a b - - - r - c d t h j k - - l m o p
a b - - - r - c d t h j k - - l m n p
a b - - - r - c d t h j - - - l m n p
a b - - - r - c d t h j - - - l m o p
a b - - - r - c d t h - - - - l m n p
a b - - - r - c d t h - - - - l m o p
a b - - - r d c - t h - k j - l m n p
a b - - - r d c - t h - k j - l m o p
a b - - - r d c - t h - - j - l m o p
a b - - - r d c - t h - - j - l m n p
a b - - - r d c - t h j k - - l m n p
a b - - - r d c - t h j k - - l m o p
a b - - - r d c - t h - k - - l m o p
a b - - - r d c - t h - k - - l m n p
a b - - - r d c - t h - - - - l m n p
a b - - - r d c - t h - - - - l m o p
a b e f g - - - - - h - - j k l m o p
a b e f g - - - - - h - - j k l m n p
a b e f - - - - - - h - - j k l m o p
a b e f - - - - - - h - - j k l m n p
a b e - - - - - - - h - - j k l m n p
a b e - - - - - - - h - - j k l m o p
a b e f g - - - - - h - k j - l m n p
a b e f g - - - - - h - k j - l m o p
a b e f - - - - - - h - k j - l m o p
a b e f - - - - - - h - k j - l m n p
a b e f g - - - - - h - - j - l m o p
a b e f g - - - - - h - - j - l m n p
a b e f - - - - - - h - - j - l m n p
a b e f - - - - - - h - - j - l m o p
a b e f g - - - - - h - k - - l m n p
a b e f g - - - - - h - k - - l m o p
a b e f - - - - - - h - k - - l m o p
a b e f - - - - - - h - k - - l m n p
a b e f g - - - - - h - - - - l m o p
a b e f g - - - - - h - - - - l m n p
a b e f - - - - - - h - - - - l m n p
a b e f - - - - - - h - - - - l m o p
a b e - - - - - - - h - k j - l m n p
a b e - - - - - - - h - k j - l m o p
a b e - - - - - - - h - k - - l m n p
a b e - - - - - - - h - k - - l m o p
a b e - - - - - - - h - - j - l m o p
a b e - - - - - - - h - - j - l m n p
a b e - - - - - - - h - - - - l m o p
a b e - - - - - - - h - - - - l m n p

a b e f g r d c d t h j k j k l m n p
```

Figure 4.15: MTA after cd_3 (corresponding to model m_3).

and remaining traces are clustered with the help of AHC. Each cluster is aligned to produce a consensus trace. The sub log and the cluster number of each consensus



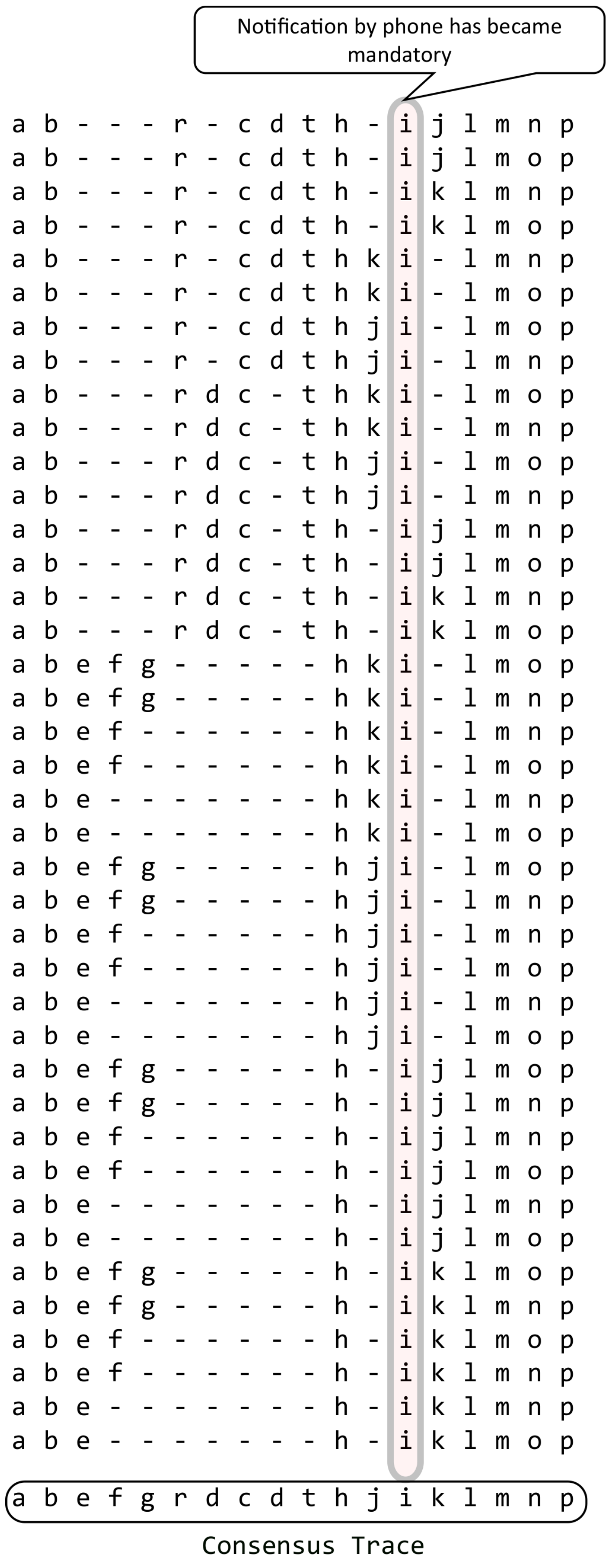
```
a b r d c t h - - - i k j l m o p
a b r d c t h - - - i k j l m n p
a b r d c t h - - - i k j l m n p
a b r d c t h - - - i - j l m o p
a b r d c t h - - - i - j l m n p
a b r d c t h - j k i - - l m n p
a b r d c t h - j k i - - l m o p
a b r d c t h - j - i - - l m n p
a b r d c t h - j - i - - l m o p
a b r d c t h - - - i - - l m n p
a b r d c t h - - - i - - l m o p
a b r d c t h - j - - - - l m o p
a b r d c t h - j - - - - l m n p
a b r d c t h - - - - - - l m n p
a b r d c t h - - - - - - l m o p
a b r d c t h k - - i - j l m n p
a b r d c t h k - - i - j l m o p
a b r d c t h k - - i - - l m o p
a b r d c t h k - - i - - l m n p
a b r d c t h k j - i - - l m n p
a b r d c t h k j - i - - l m o p
a b r d c t h k j - - - - l m o p
a b r d c t h k j - - - - l m n p
a b r d c t h - j - i k - l m o p
a b r d c t h - j - i k - l m n p
a b r d c t h - - - i k - l m n p
a b r d c t h - - - i k - l m o p
a b r d c t h i j - - k - l m n p
a b r d c t h i j - - k - l m o p
a b r d c t h - j - - k - l m o p
a b r d c t h - j - - k - l m n p
a b r d c t h - - - - k - l m n p
a b r d c t h - - - - k - l m o p

a b r d c t h k j k i k j l m n p
```

Figure 4.16: MTA after cd_4 (corresponding to model m_4).

trace are recorded, this information will be used during the drift localization.

Clustering and aligning each of 480 sub logs of insurance claim process has resulted in total 1440 consensus traces. *Information score, activity relationship*, and *relation entropy* feature sets for each consensus trace is calculated. The successive population of the feature are selected using windowing instance selection procedure and investigated for concept drift by applying *statistical hypothesis testing* procedures.

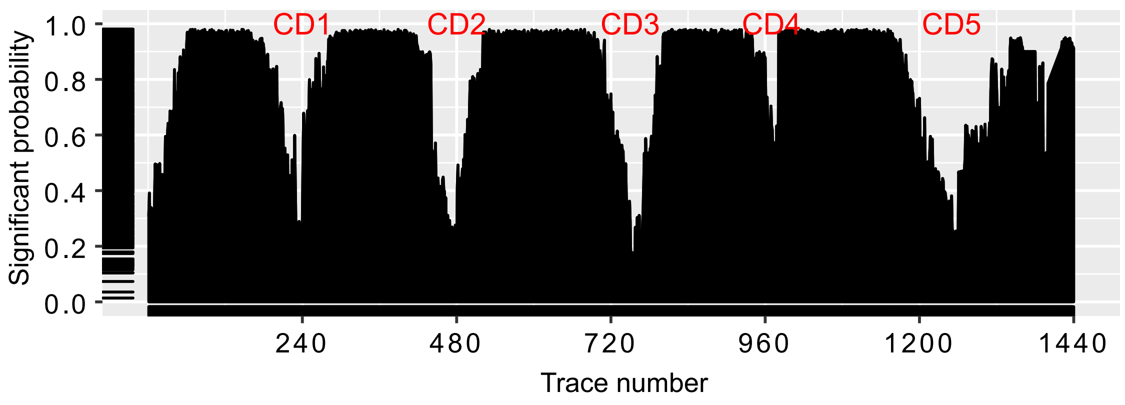
The graphs shown in fig. 4.18 illustrate the result of detecting concept drifts



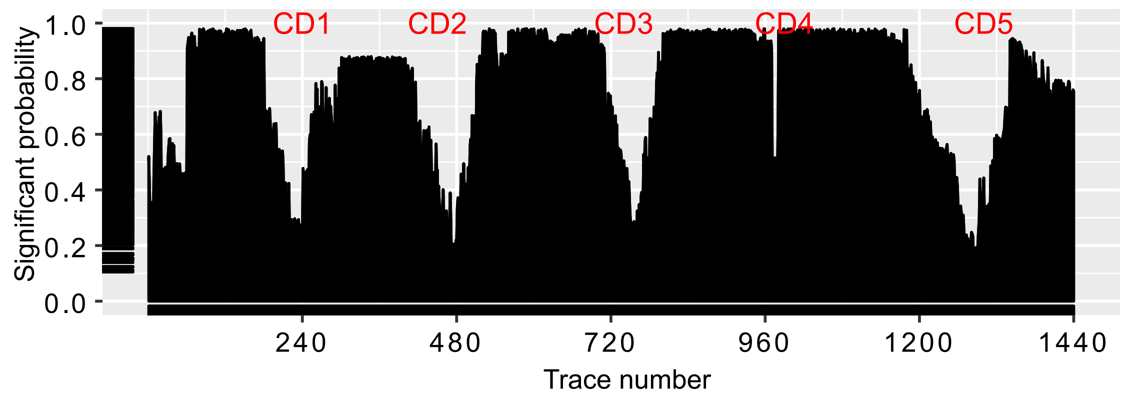
```
a b - - - r - c d t h - i j l m n p
a b - - - r - c d t h - i j l m o p
a b - - - r - c d t h - i k l m n p
a b - - - r - c d t h - i k l m o p
a b - - - r - c d t h k i - l m n p
a b - - - r - c d t h k i - l m o p
a b - - - r - c d t h j i - l m o p
a b - - - r - c d t h j i - l m n p
a b - - - r d c - t h k i - l m o p
a b - - - r d c - t h k i - l m n p
a b - - - r d c - t h j i - l m o p
a b - - - r d c - t h j i - l m n p
a b - - - r d c - t h - i j l m n p
a b - - - r d c - t h - i j l m o p
a b - - - r d c - t h - i k l m n p
a b - - - r d c - t h - i k l m o p
a b e f g - - - - - h k i - l m o p
a b e f g - - - - - h k i - l m n p
a b e f - - - - - - h k i - l m n p
a b e f - - - - - - h k i - l m o p
a b e - - - - - - - h k i - l m n p
a b e - - - - - - - h k i - l m o p
a b e f g - - - - - h j i - l m o p
a b e f g - - - - - h j i - l m n p
a b e f - - - - - - h j i - l m n p
a b e f - - - - - - h j i - l m o p
a b e - - - - - - - h j i - l m n p
a b e - - - - - - - h j i - l m o p
a b e f g - - - - - h - i j l m o p
a b e f g - - - - - h - i j l m n p
a b e f - - - - - - h - i j l m n p
a b e f - - - - - - h - i j l m o p
a b e - - - - - - - h - i j l m n p
a b e - - - - - - - h - i j l m o p
a b e f g - - - - - h - i k l m o p
a b e f g - - - - - h - i k l m n p
a b e f - - - - - - h - i k l m o p
a b e f - - - - - - h - i k l m n p
a b e - - - - - - - h - i k l m n p
a b e - - - - - - - h - i k l m o p

a b e f g r d c d t h j i k l m n p
```

Figure 4.17: MTA after cd_5 (corresponding to model m_5).

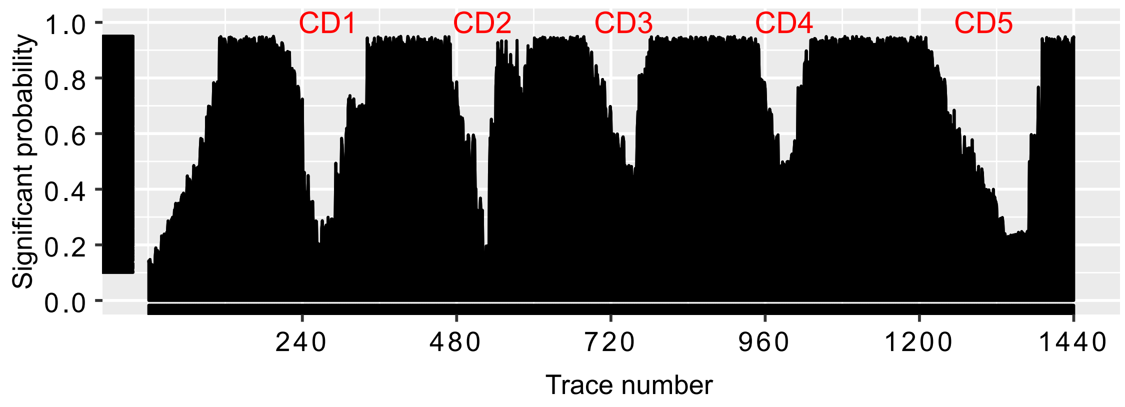
around 240th, 480th, 720th, 960th and 1200th consensus traces (continuous dip in *significant probability* indicates the change in process). The 240th consensus trace is



(a) Result of detecting concept drift by applying *Siegel Tukey* test on information score feature.



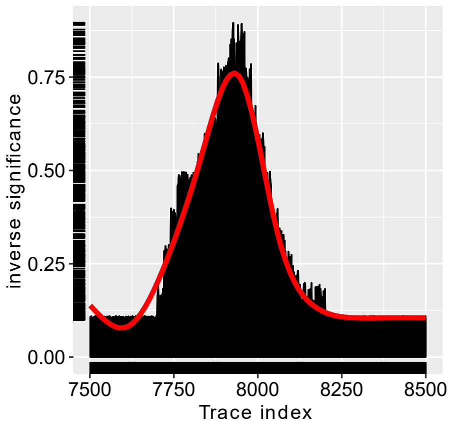
(b) Result of detecting concept drift by applying *Chi-square* test on relationship count feature.



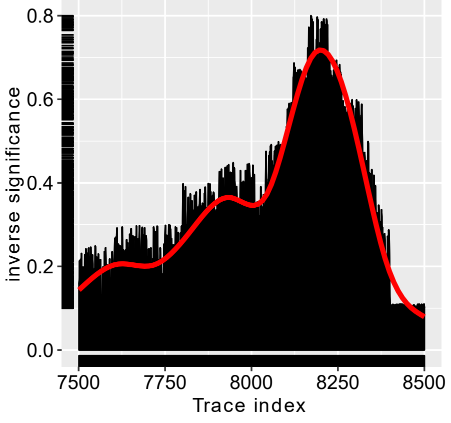
(c) Result of detecting concept drift by applying *Chi-square* test on relationship entropy feature.

Figure 4.18: Result of concept drift detection.

derived by aligning the traces of 80th sub log; this indicates process change around the 8000th trace.



(a) Change localization at activity `by post`.



(b) Change localization at activity `by phone`.

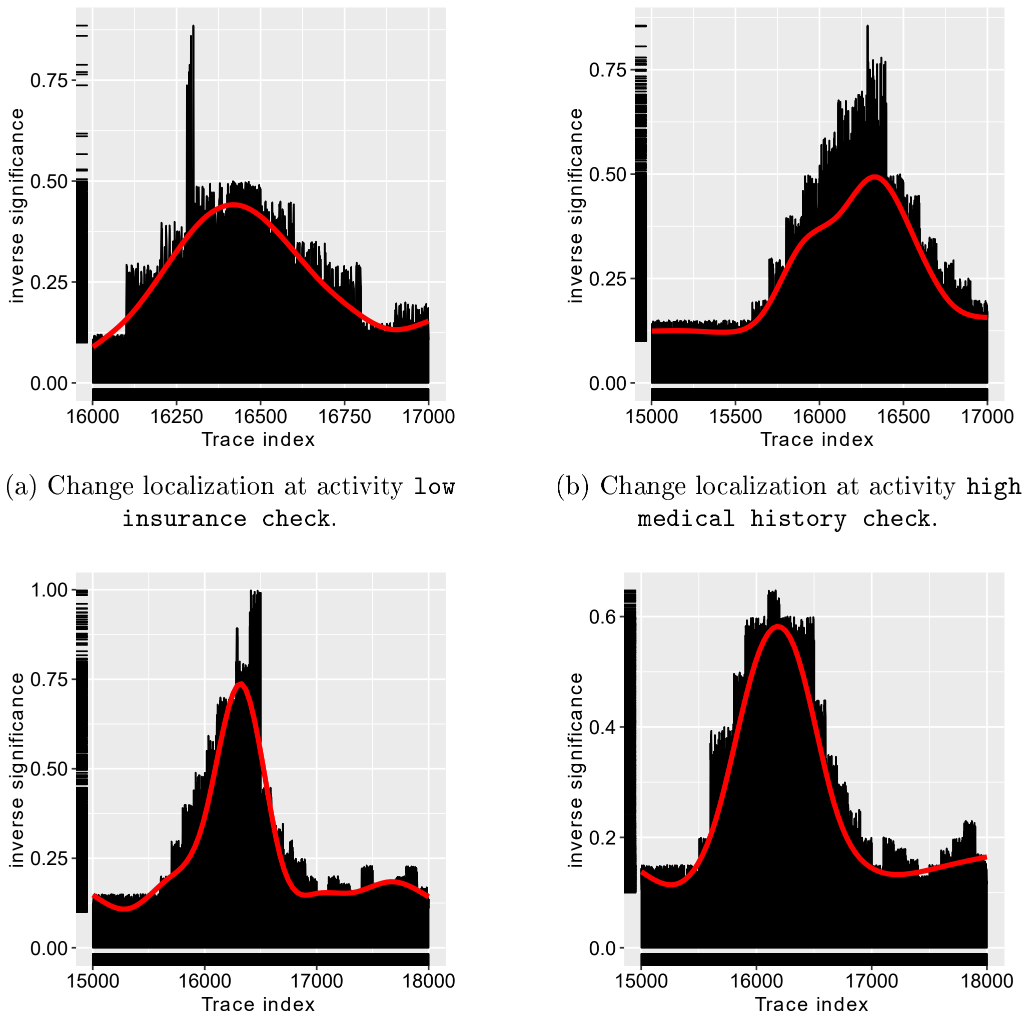
Figure 4.19: Concept drift localization around trace index $8,000$.

Based on the evidence shown in fig. 4.18, we arrive at the conclusion that there is concept drifts around the 8000^{th}, 16000^{th}, 24000^{th} and 40000^{th} trace indexes corresponding to the sub logs.

The result of concept drift detection shown in fig. 4.18 on all three feature sets (information score, relationship count, and relation entropy) failed to legitimately capture the concept drift cd_4 (around 960^th consensus trace). This is due to the characteristics of hypothesis tests used in detecting the concept drift.

After detecting the concept drift, 80^{th}, 160^{th}, 240^{th}, 320^{th}, and 400^{th} sub logs are further investigated to localize it. Concept drift localization techniques are used for identifying the changes at the level of activities, relationship between activities, and traces. Trace specific feature sets are generated by using window count method on each of the traces of 80^{th}, 160^{th}, 240^{th}, 320^{th} and 400^{th} sub logs by considering every pair of activities. Successive populations of window count values are compared with the help of hypothesis testing procedures.

The graph shown in fig. 4.19 illustrates the concept drift localization corresponding to activity `by post` (shown in fig. 4.19a) and `by phone` (shown in fig 4.19b) of the process shown in fig. 4.8, around the traces corresponding to case id $8,000$ by applying *Chi-square* test on window count feature around. Occurrence of this concept drift arranges activities `by post`, `by email`, and `by phone` parallel between AND split/join from XOR split/join. Which enables to execute all of these activities simultaneously



(a) Change localization at activity `low insurance check`.

(b) Change localization at activity `high medical history check`.

(c) Change localization at activity `prepare notification`.

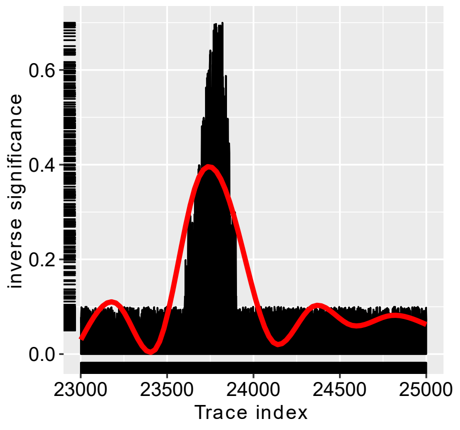
(d) Change localization at activity `by email`.

Figure 4.20: Concept drift localization around trace index 16, 000.

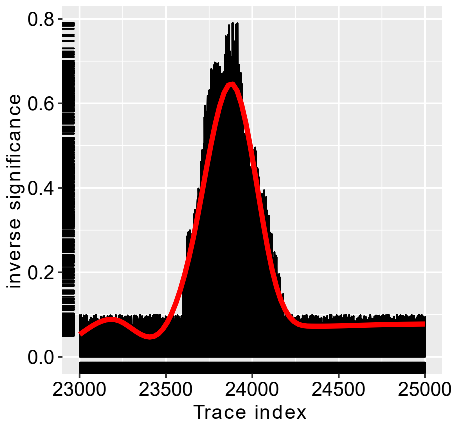
in any order.

The graph shown in fig. 4.20 illustrates the concept drift localization corresponding to activity `low insurance check` (shown in fig. 4.20a), `high medical history check` (shown in fig 4.20b), `prepare notification` (shown in fig. 4.20c), and `by email` (shown in fig 4.20d) of the process shown in fig. 4.9a, around traces corresponding to case id 16, 000, it is obtained by applying *Chi-square* test on window count feature. This concept drift is primarily due to the result of arranging high/low insurance claim tasks parallel and replacing AND split/join with XOR split/join at prepare notification activity.

The graph shown in fig. 4.21 illustrates concept drift localization corresponding to



(a) Change localization at activity `high claim join`.



(b) Change localization at activity `prepare notification`.

Figure 4.21: Concept drift localization around trace index $24,000$.

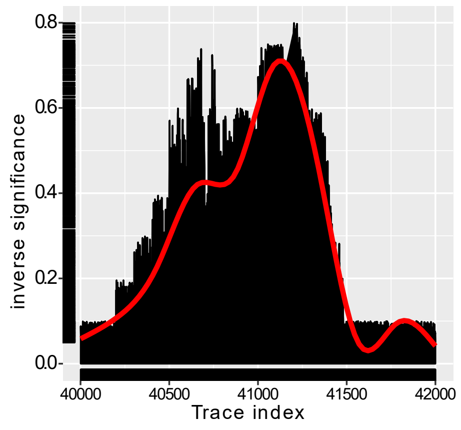
activity `high claim join` (shown in fig. 4.21a) and `prepare notification` (shown in fig 4.21b) of the process shown in fig. 4.9b. It is around the traces corresponding to case id $24,000$. It is obtained by applying *Chi-square* test on window count feature around trace index $24,000$. This concept drift resulted due to introduction of knock-out structure at high insurance claim path and replacing XOR split/join by OR split/join at `prepare notification`.

The graph shown in fig. 4.22 illustrates concept drift localization corresponding to activity `by post` (shown in fig. 4.22a), `by email` (shown in fig 4.22b), `by phone` (shown in fig. 4.22c), and `prepare notification` (shown in fig 4.22d) of the process shown in fig. 4.9d around the traces corresponding to case id $40,000$. It is obtained by applying *Chi-square* test on window count feature around trace index $40,000$.

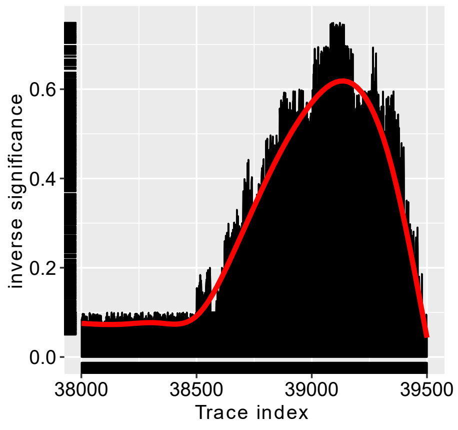
The localization result about the concept drift detection at 320^{th} sub log (around case id $32,000$) is not shown. The *window count* feature used in this chapter has failed to localize the process change (significance probability did not show any changes).

4.7 Comparison Results

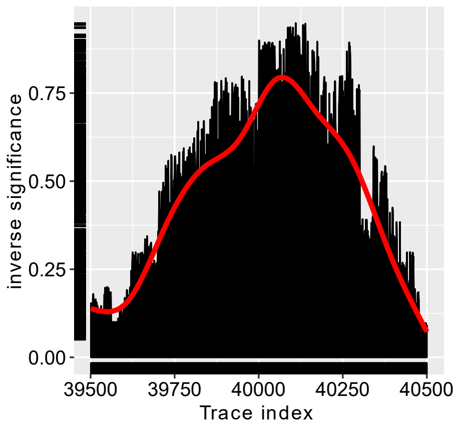
This section demonstrates the result of the comparison study between existing and the proposed method for concept drift detection and localization. Feature sets used for handling concept drift (in existing and proposed method) are evaluated against the time required for detecting and localizing concept drift (both in cumulative and



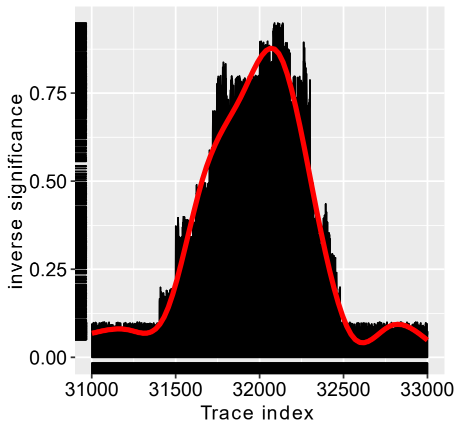
(a) Change localization at activity `by post`.



(b) Change localization at activity `by email`.



(c) Change localization at activity `by phone`.



(d) Change localization at activity `prepare notification`.

Figure 4.22: Concept drift localization around trace index 40, 000.

non-cumulative manner). Demonstrated results are the average over five independent runs of algorithms (error bars are shown in corresponding graphs).

4.7.1 Time taken for detecting concept drift

Comparison results are shown in fig. 4.23-4.24 illustrates the time required for individually detecting five different concept drifts by applying our method and the technique suggested by Bose et al. (2011). It evaluates the efficiency of various feature sets and technique of proposed and existing methods (demonstrates that how quick the feature sets are capable of detecting concept drift).

The cumulative time required for detecting five different concept drifts are pre-

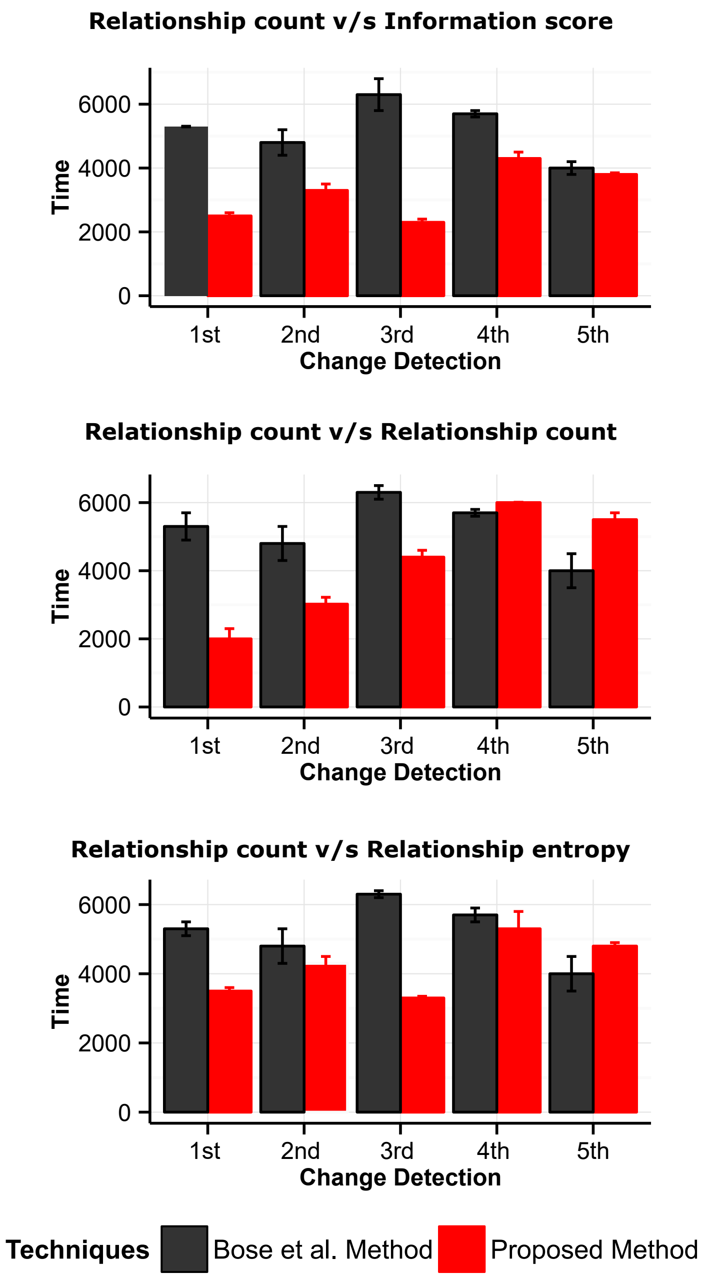


Figure 4.23: Time taken for detecting individual concept drifts (Relationship count v/s proposed features).

sented in the fig. 4.25-4.26. It illustrates the total time needed for detecting all five concept drift in a process by applying our method and the method proposed by Bose et al. (2011). Results presented in fig. 4.23-4.26 clearly shows that the proposed method is fast and responsive in handling concept drift.

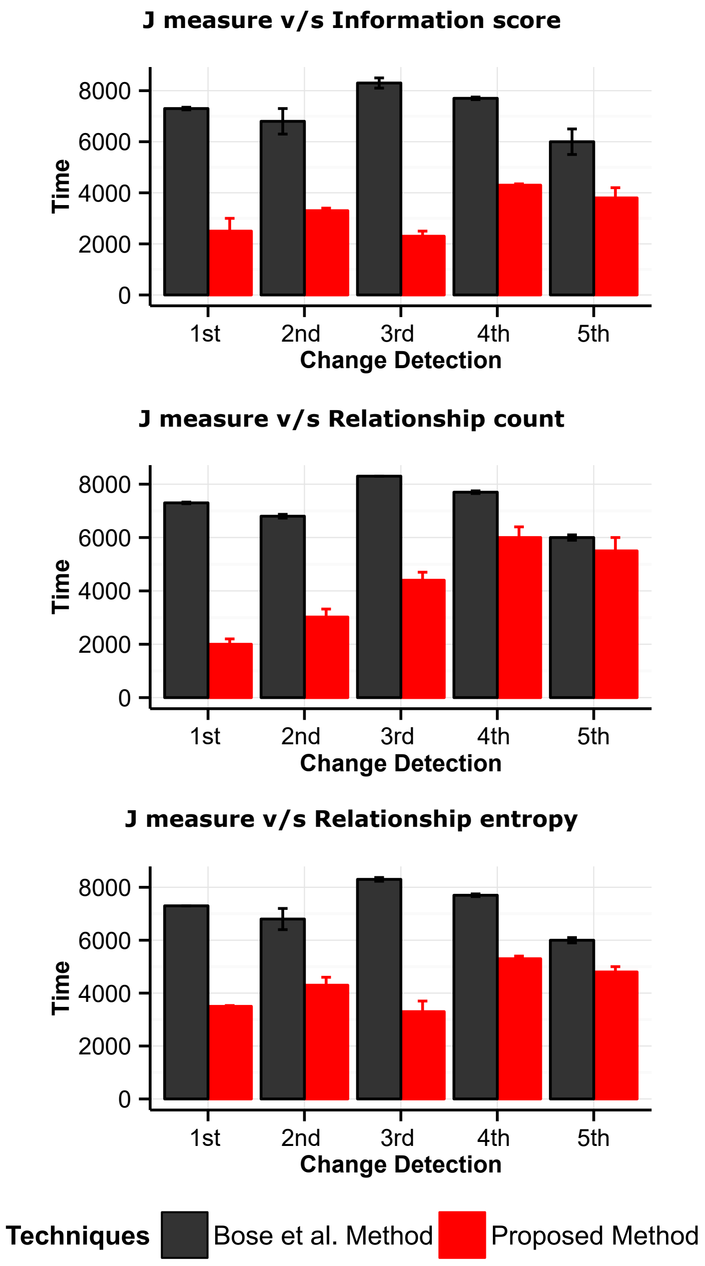


Figure 4.24: Time taken for detecting individual concept drifts (J measure v/s proposed features).

4.7.2 Receiver operating characteristic curves

The Receiver operating characteristic (ROC) curve (plotted with sensitivity versus 1-specificity) performs the comparison of different approaches for detecting and localizing concept drifts and chooses the best approach. An Area Under Curve (AUC) =1 is known as perfect discrimination, and 0.5 is known as absence of discrimination. An AUC was calculated using all detection and localization attempts, and for each attempt, the success and failure are recorded based on the result.

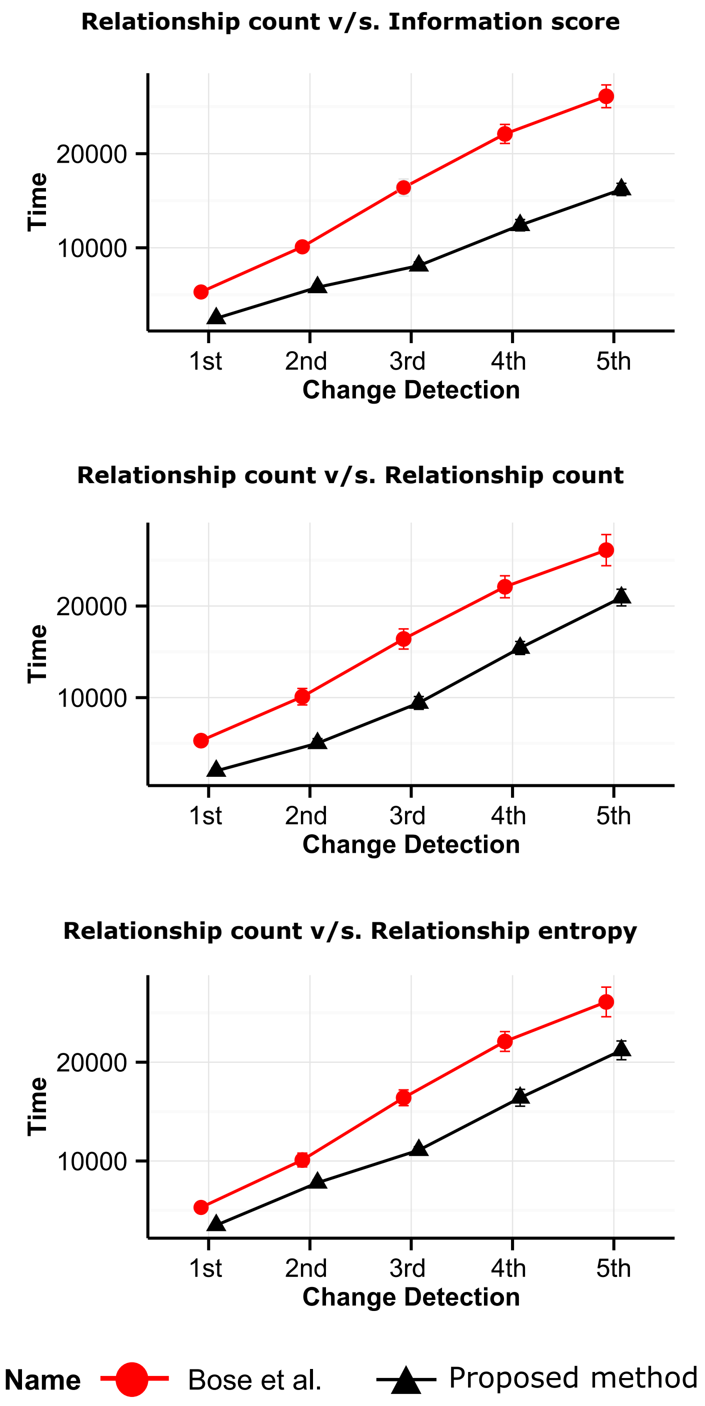


Figure 4.25: Cumulative time taken for detecting concept drifts (Relationship count v/s proposed features).

ROC curve for demonstrating the AUC of concept drift detection of the proposed and other methods is shown in fig. 4.27 and 4.28. AUC of proposed method (0.9728) is higher when compared to the AUC of Bose et al. (2014) (0.9439) and Carmona and Gavalda (2012) (0.7236). The greater AUC demonstrates that the proposed approach is capable of sensitively and correctly detecting concept drift.

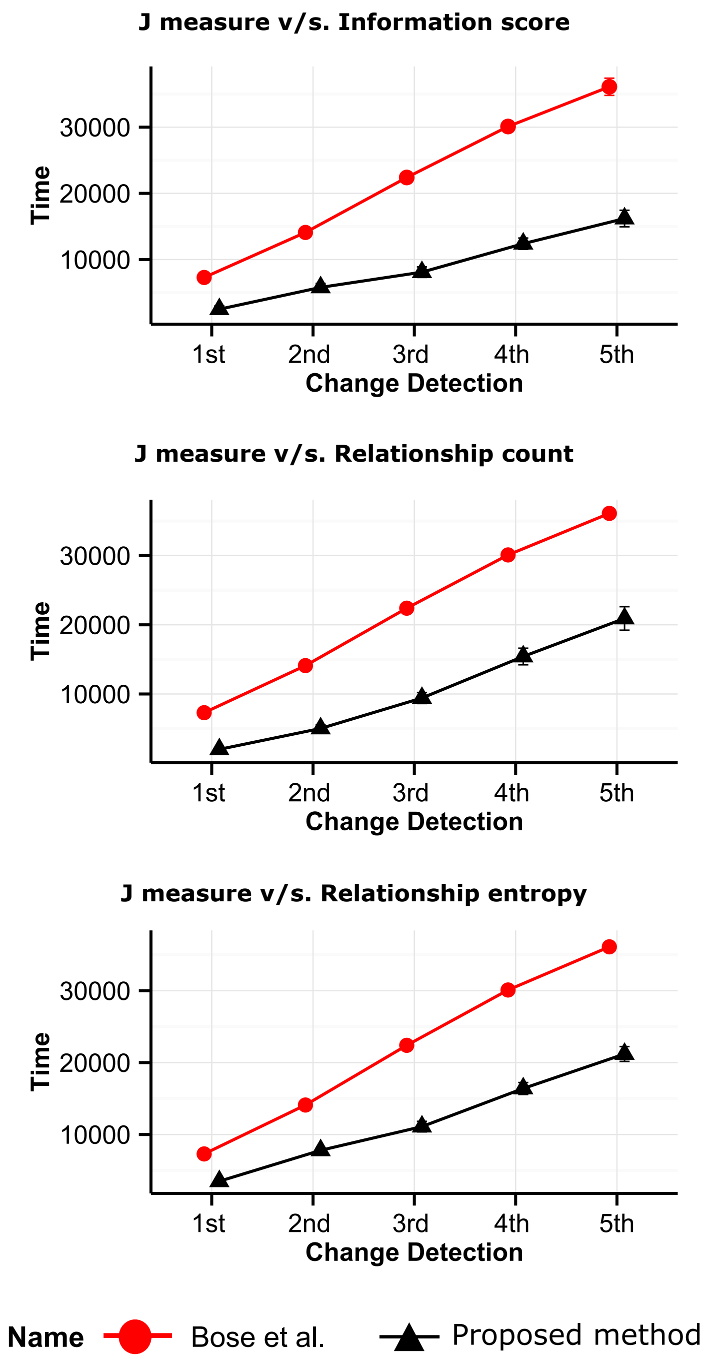


Figure 4.26: Cumulative time taken for detecting concept drifts (J measure v/s proposed features).

ROC curves for concept drift localization is shown in fig. 4.29. AUC of proposed method (0.9036) is significantly higher than the method proposed by Bose et al. (2014) (0.8257). Based on the comparison of AUCs of the different methods used for concept drift detection and localization, we can arrive at the conclusion that the proposed method is capable of handling the situation of concept drift efficiently.

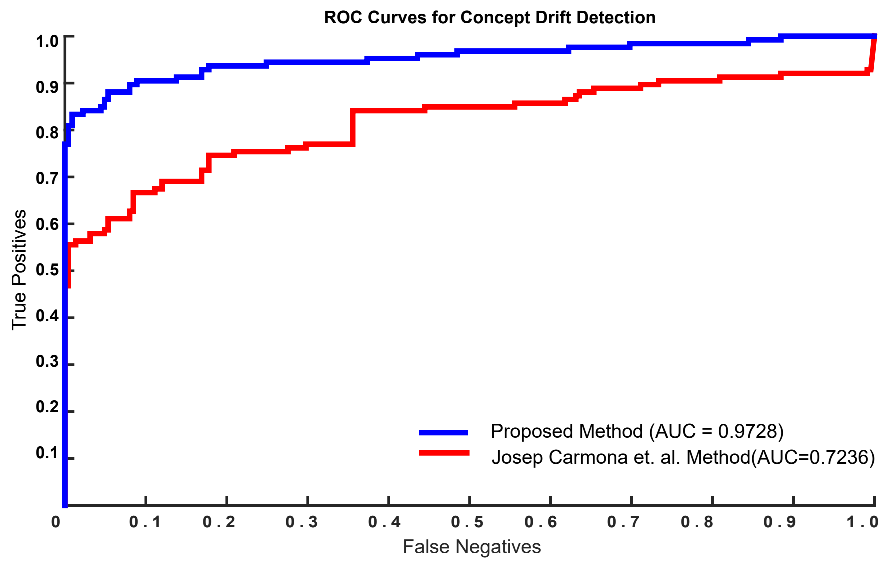


Figure 4.27: ROC curve for detecting concept drift (Proposed method vs. Bose et al. (2011)).

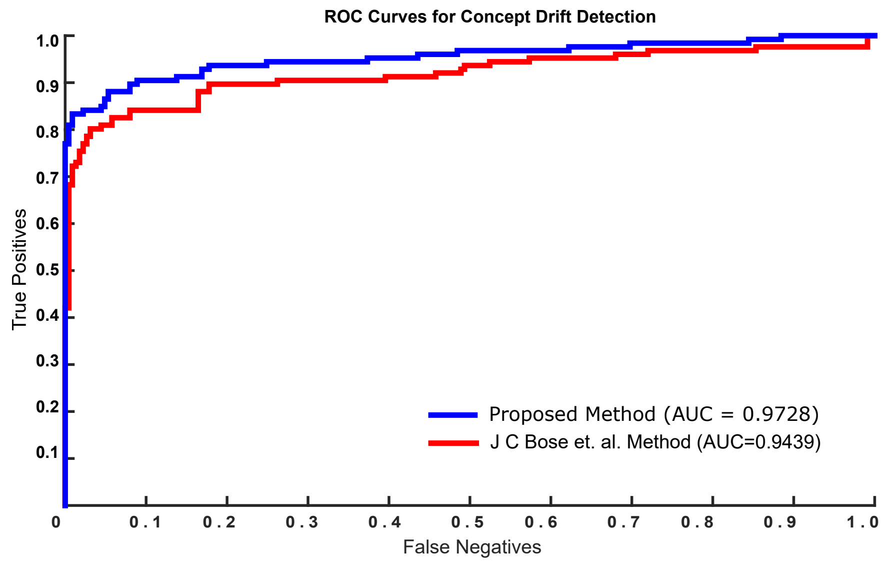


Figure 4.28: ROC curve for detecting concept drift (Proposed method vs. Carmona and Gavalda (2012)).

4.8 Summary

In process mining, concept drift is a condition wherein process changes during the period of execution/analysis. Handling concept drifts efficiently is highly crucial for ensuring the validity and correctness of the results generated by process mining. Concept drift in the process makes the result of analysis invalid. The facets from which the concept drift phenomenon to be addressed are many. One needs to consider every major *change type*, *perspective*, *pattern of change* and *mode of handling* to propose a

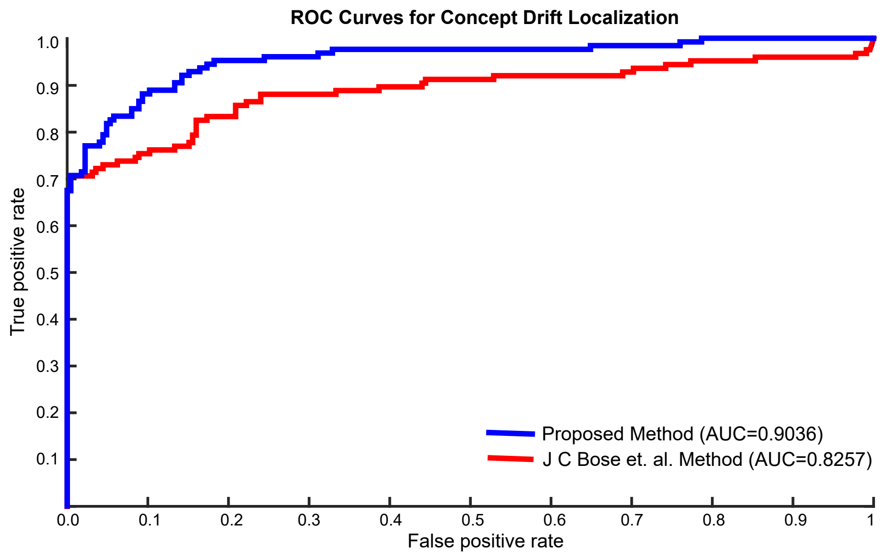


Figure 4.29: ROC curve for localizing localizing drift (Proposed method vs. Bose et al. (2011)).

end-to-end solution.

This chapter has demonstrated a method for *detecting* and *localizing* concept drift in *control-flow* perspective. Consensus traces obtained by MTA are processed to extract the control-flow features. A systematic examination of feature values with application of statistical hypothesis testing techniques assists in detecting and localizing concept drift.

It is demonstrated that the proposed method is capable of efficiently uncovering most of the change types related to control-flow perspective in the *single run* of the algorithm using a reduced feature set size.

Chapter 5

Process Logo: Informative Visualization of Control Flow Perspective

For building a control flow process model, there are a plethora of techniques exist in process mining. Currently, the available control flow discovery algorithms in process mining are capable of generating process model similar to the one shown in fig. 5.1 (constructed using YAWL notation). From this category of a process model, no information other than activities (steps) and path (connections between steps) can be depicted.

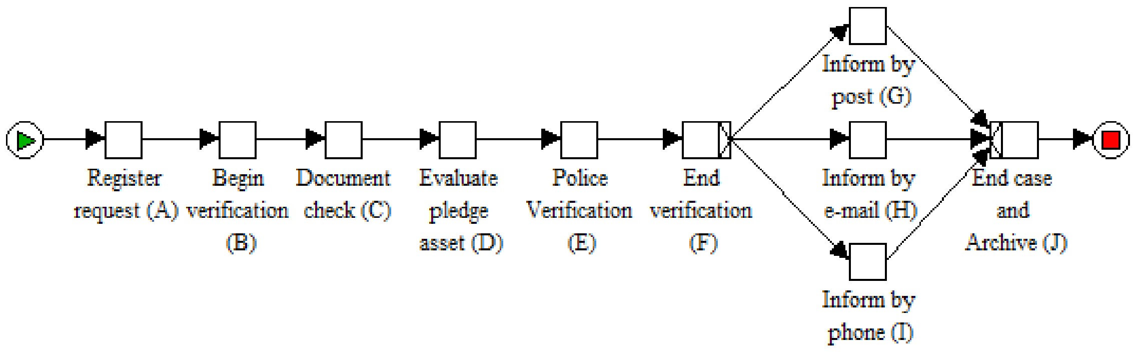


Figure 5.1: Control-flow of loan application handling process (initial variant).

This chapter introduces a new control flow modeling notation called process logo. Process logo is capable of depicting following a set of information in a single compact graphic (in addition to the conventional causal relationships.)

- Order of occurrence of activities (causal relationship).

- General consensus between activities (back bone of control flow).
- Co-occurrence of related activities at any given position in trace.
- Information score of individual activity.
- Conserved and shared activities/sequences.

A process logo is created from a set of *aligned traces* generated using MTA, basically consists of a stack of activities at each position. The relative size of the activity indicates its frequency and height depicts the information content of the position at which the activity (or set of activities) is observed (in bits).

The upcoming contents in this chapter is organized as follows, section 5.1 presents the framework followed for constructing process logo. An example of MTA illustrating different splits and joins is given in section 5.2. Features required for constructing process logo is given in section 5.3. Results of experiments carried out is given in section 5.4 and the chapter concludes with the concise summary.

5.1 Framework

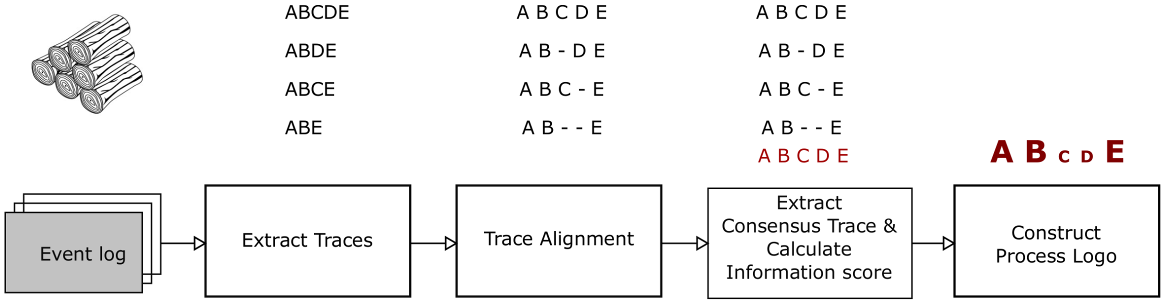
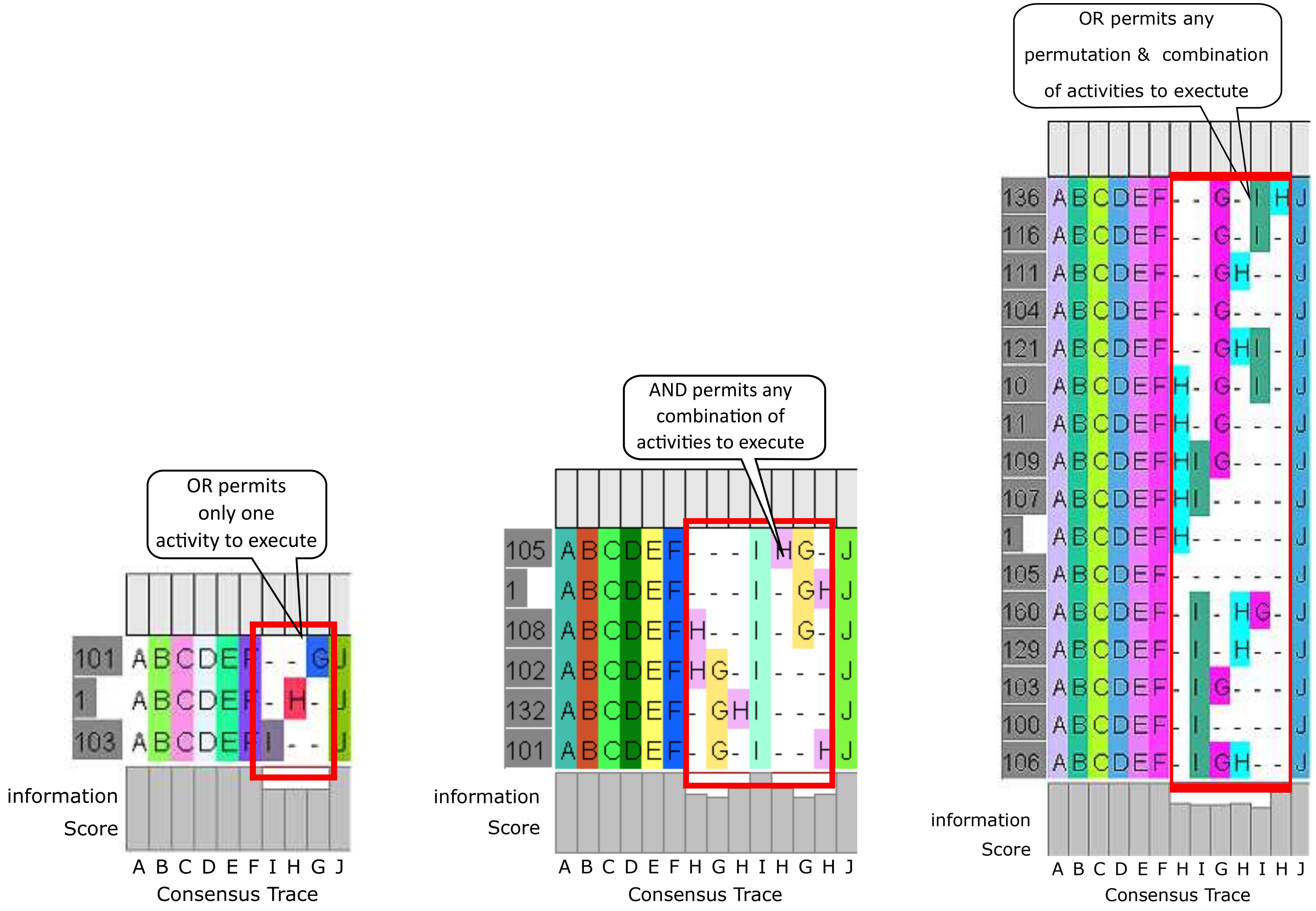


Figure 5.2: Framework for constructing process logo.

Framework followed to construct process logo is shown in fig. 5.2. Initially, all traces in the event log are extracted and duplicates are filtered out to form a distinct set of traces. Alignment between traces is generated with the help of MTA. It is used to extract consensus trace and information score. Finally, a process logo representing control flow of the process is generated.

Table 5.1: Event log of loan application handling process.

Case ID	Event ID	Activity	Resource	Timestamp
ab12	2342	Register request (`A`)	r71324	10-10-2014/01:20
	2343	Begin verification (`B`)	r21345	10-10-2014/03:23
	2344	Document check (`C`)	r67890	10-10-2014/05:25
	2345	Evaluate pledge assets (`D`)	r45634	11-10-2014/12:20
	2346	Police verification (`E`)	r55467	11-10-2014/13:44
	2347	End verification (`F`)	r34526	14-10-2014/15:20
	2348	Inform by post (`G`)	r22435	16-10-2014/04:55
	2349	End case and archive (`J`)	r11234	19-10-2014/01:47
ab13	3313	Register request (`A`)	r71324	28-12-2014/13:44
	3314	Begin verification (`B`)	r21345	29-12-2014/14:23
	3315	Document check (`C`)	r67890	30-12-2014/17:01
	3316	Evaluate pledge assets (`D`)	r45634	01-01-2015/13:55
	3317	Police verification (`E`)	r55467	01-01-2015/16:43
	3318	End verification (`F`)	r34526	01-01-2015/19:52
	3319	Inform by e-mail (`H`)	r22434	02-01-2015/13:34
	3320	End case and archive (`J`)	r11234	08-01-2015/14:52
ab14	4313	Register request (`A`)	r71324	28-12-2014/13:44
	4314	Begin verification (`B`)	r21345	29-12-2014/14:23
	4315	Document check (`C`)	r67890	30-12-2014/17:01
	4316	Evaluate pledge assets (`D`)	r45634	01-01-2015/13:55
	4317	Police verification (`E`)	r55467	01-01-2015/16:43
	4318	End verification (`F`)	r34526	01-01-2015/19:52
	4319	Inform by phone (`I`)	r22434	02-01-2015/13:34
	4320	End case and archive (`J`)	r11234	08-01-2015/14:52
	...	...	...	...


OR permits only one activity to execute
information Score
A B C D E F I H G J
Consensus Trace
AND permits any combination of activities to execute
information Score
A B C D E F H G H I H G H J
Consensus Trace
OR permits any permutation & combination of activities to exectute
information Score
A B C D E F H I G H I H J
Consensus Trace

(a) XOR (b) AND (c) OR

Figure 5.3: MTAs of variants of loan application handling process.

We use variants of loan application handling process (shown in fig. 5.1) as a running example for illustrating the concepts discussed in this chapter. Excerpt of the event log corresponding to the process is shown in the table 5.1. Materials and methods presented in this chapter are strictly related to control flow perspective, therefore activity column in the event log is the bare minimum requirement for constructing the process logo (all other information except activity column is filtered out).

We assign a distinct name for every activity of loan process to refer them easily. These short names can be found corresponding to every activity in the process model. All process models given in this chapter are constructed using YAWL notations.

Table 5.2: Possible traces between activities `G`, `H`, and `I`.

Split/ join type	**Possible traces**
Serial	$\mathcal{A} = \{$ `ABCDEFGHIJ` $\}$
XOR	$\mathcal{B} = \{$ `ABCDEFGJ, ABCDEFHJ, ABCDEFIJ` $\}$
OR	$\mathcal{C} = \{$ `ABCDEFGIJH, ABCDEFIGHJ, ABCDEFIHGJ, ABCDEFHIGJ, ABCDEFHGIJ, ABCDEFGHJ, ABCDEFIJ` `ABCDEFHGJ, ABCDEFGIJ, ABCDEFIGJ, ABCDEFIHJ, ABCDEFHIJ, ABCDEFGJ, ABCDEFHJ, ABCDEFGHIJ`
AND	$\mathcal{D} = \{$ `ABCDEFGHIJ, ABCDEFGIJH, ABCDEFIGHJ, ABCDEFIHGJ, ABCDEFHIGJ, ABCDEFHGIJ` $\}$

5.2 An Example of Arranging Traces Using MTA

An example of aligning traces related to variants of loan application handling process is illustrated in this section. Causal relationships between activities `G`, `H`, and `I` of loan application process are altered to create variants. MTA of each process variant is extracted and its competence in capturing activity relationship pattern is demonstrated.

Examples of aligning three different variants of loan application process are shown in fig. 5.3. Corresponding to each MTA, the left panel shows case ids as in the event log (gray background indicate traces that have identical duplicates), each column in the MTA represents set of activities, and each row in the alignment represents a distinct trace. The bottom panel depicts the information score for each column and consensus trace. Three variants of loan application process is described as follows,

- According to fig. 5.1, activities `G`, `H` and `I` ordered between XOR split/join. This activity arrangement permits execution of any one of the three activities to execute at any given instance (possible traces are given in set $\mathcal{B}$ of table 5.2). Corresponding MTA is shown in fig. 5.3a.

- Set $\mathcal{C}$ in table 5.2 is result of ordering activities `G`, `H`, and `I` between AND split/join. This permits execution of these three activities in any permutation. Corresponding MTA is shown in fig. 5.3b.

- Set $\mathcal{D}$ in table 5.2 is the result of ordering activities `G`, `H` and `I` between OR split/join. This permits the execution of activities in any permutation and combination. The corresponding MTA is shown in fig. 5.3c.

From the alignments shown in fig. 5.3 it is evident that MTA is capable of legibly capturing the prominent activity arrangements (such as serial, OR, XOR, AND etc). These alignments are used to extract the information related for generating process logo.

5.3 Features for Constructing Process Logo

The information necessary for constructing process logo is discussed in this section.

5.3.1 Consensus trace

Consensus trace generally consists of sequence of activities. Each activity in consensus trace corresponds to significant activity at each column (position) of MTA. Consensus trace captures the backbone of control-flow and prevalence (or order) of activity arrangements.

Fig. 5.3 shows consensus traces highlighted corresponding to each MTA (`ABCEDHGIF`, `ABCDFGHGIHGHJ` and `ABCEDFGHIGHIJ`). To obtain a consensus trace, a table representing frequencies of activities in every column of MTA is constructed and frequency value is sorted in decreasing order. Finally, an activity with the highest frequency is chosen to appear on consensus trace.

5.3.2 Information score

Information score (R) is used to represent the height of activities(y-axis) in process logo. It is calculated using equation 5.1.

$$R_i = \log_2(|\Sigma|) - (H_i + e_n) \tag{5.1}$$

Where, H_i is the uncertainty (Shannon, 2001) of position i. H_i is calculated using equation 5.2, and the value of e_n in equation 5.1 is calculated using 5.3.

$$H_i = -\sum f_{\mathsf{a},i} \times \log_2 f_{\mathsf{a},i} \tag{5.2}$$

Here, $f_{\mathsf{a},i}$ is the relative frequency of activity of the process under consideration at position i, and e_n is the small-sample correction for an alignment of n letters. The approximation for e_n is calculated using equation 5.3. Where s is number of activities of the process and n is number of traces of the process.

$$e_n = \frac{1}{\ln 2} \times \frac{s-1}{2n} \tag{5.3}$$

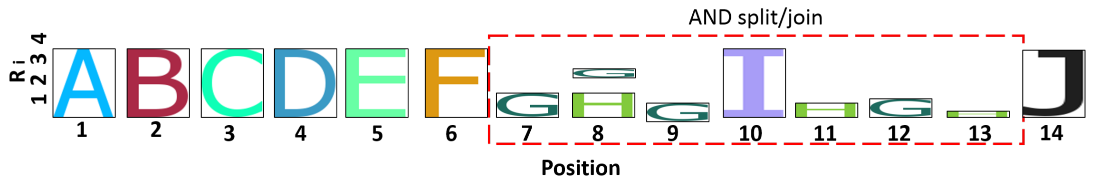
The height of a activity in column i in the process logo is calculated using equation 5.4.

$$height = f_{a,i} \times R_i \tag{5.4}$$

5.4 Results

A Process logo is constructed from the set of aligned traces. It consists of stack of activity names corresponding to each position of it's corresponding MTA. The size of activity in process logo is directly proportional to it's information score. The method for constructing process logo is applied and validated on process variants of loan application handling process. Event logs corresponding to process variants are generated using *CPN tools* (Jensen and Kristensen, 2009) with *CPNXES* (Michael, 2011) library.

Table 5.3: Attribute values of a process logo.

Position in MTA	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**		**9**	**10**	**11**	**12**	**13**	**14**
Activity	A	B	C	D	E	F	G	G	H	G	I	H	G	H	J
Frequency	200	200	200	200	200	200	102	27	102	98	200	54	98	27	200
Relative Frequency	1.0	1.0	1.0	1.0	1.0	1.0	0.51	0.13	0.51	0.49	1.0	0.26	0.49	0.13	1.0
Information Score	3.5	3.5	3.5	3.5	3.5	3.5	1.59	0.40	1.59	1.2	3.5	0.8	1.2	0.40	3.5
Consensus Trace	A	B	C	D	E	F	G	H		G	I	H	G	H	J
Process Logo															

The attributes and process logo related to MTA of fig. 5.3b are listed in table 5.3. This process logo clearly shows,

- Arrangement of activities `G`, `H` and `I` in AND split/join.
- Serial arrangement of all other activities except `G`, `H` and `I`.
- Information score of any activity in process logo is proportional to it's frequency. For example, in table 5.3, activities with frequency value equal to 200 have highest information score (3.5 bits).
- Information score decreases with increase in number of distinct activities in the any column of MTA. For example, in table 5.3 columns with only one activity have highest information score. At the same time, columns with more number of distinct activities have less information score.

Region in control flow where these activities form AND split/join is highlighted in the process logo shown in table 5.3.

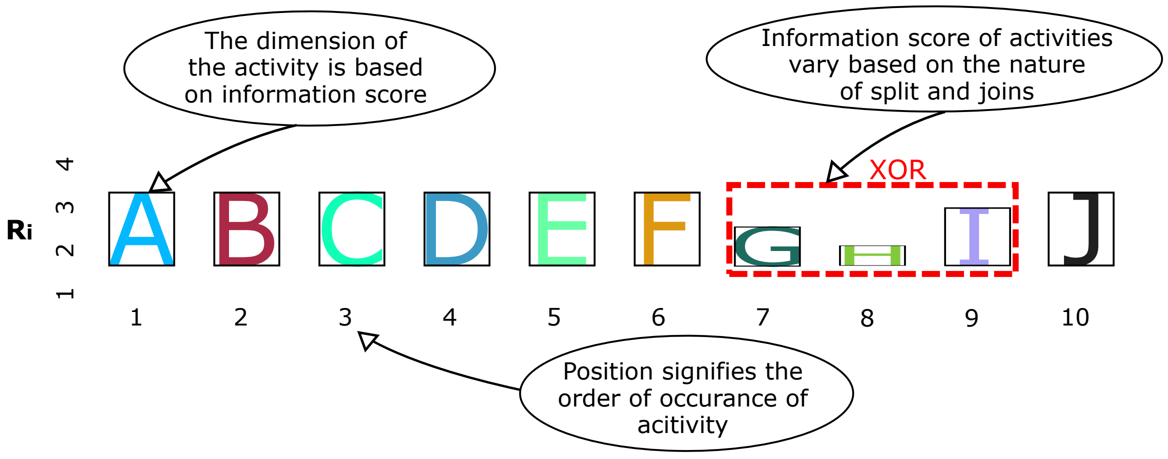
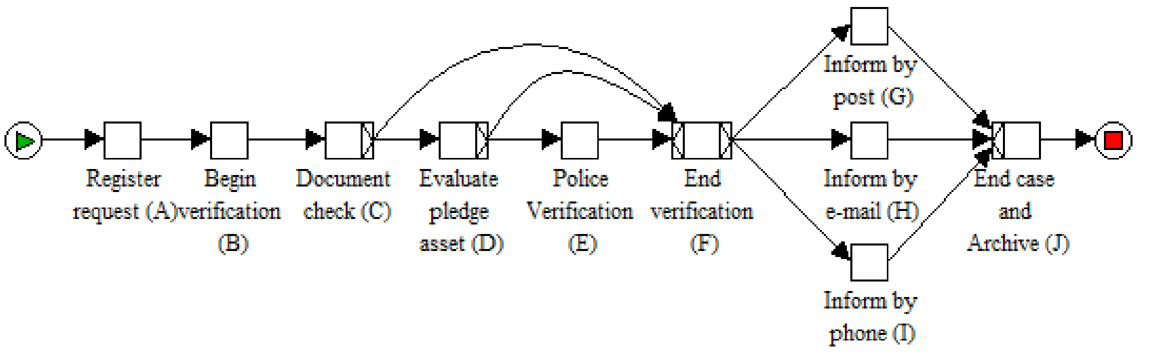

The dimension of the activity is based on information score
Information score of activities vary based on the nature of split and joins
XOR
Ri
1 2 3 4
A B C D E F G H I J
1 2 3 4 5 6 7 8 9 10
Position signifies the order of occurance of acitivity

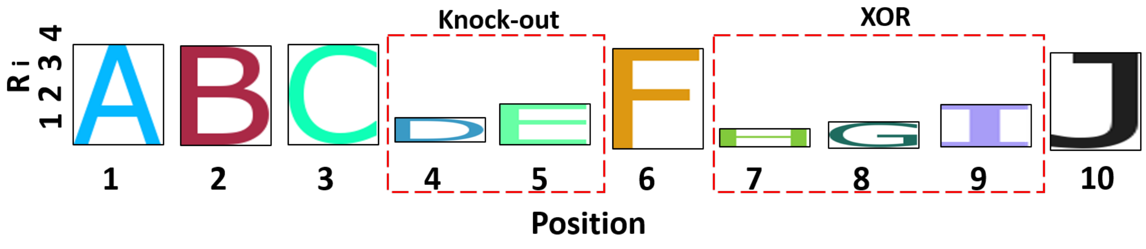
Figure 5.4: Trace logo of process having XOR split/join

Second process logo of variant of loan application process is shown in fig. 5.4. It consists of 10 distinct activities. Activities `G`, `H` and `I` are arranged between XOR split/join (all other activities except `G`, `H` and `I` are arranged serially). The XOR region consisting activities `G`, `H` and `I` is highlighted in fig. 5.4.

Third process logo is shown in fig 5.5, it corresponds to process model shown in fig. 5.5a. It consists of three activities arranged in knock out-fashion (it is the arranging

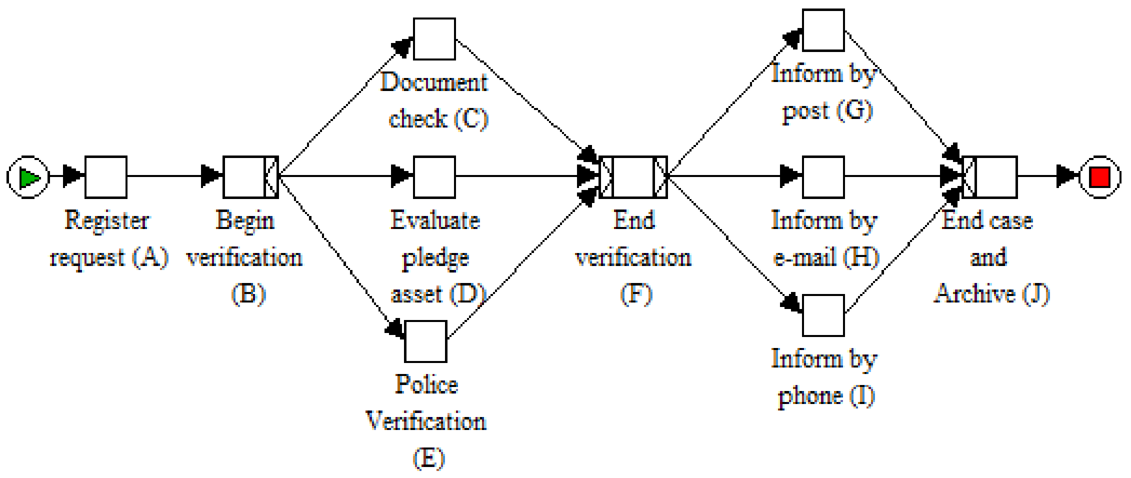


(a) Process model.

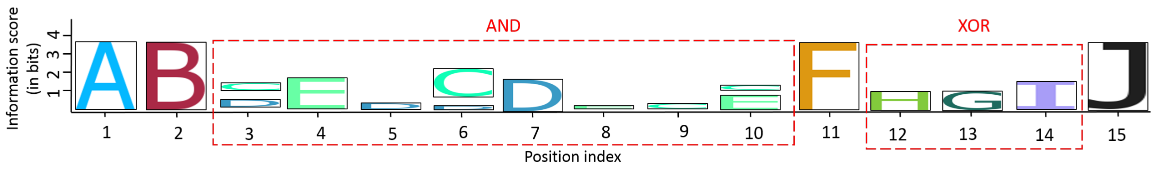


(b) Process logo.

Figure 5.5: Process logo exhibiting knock out and XOR split/join.

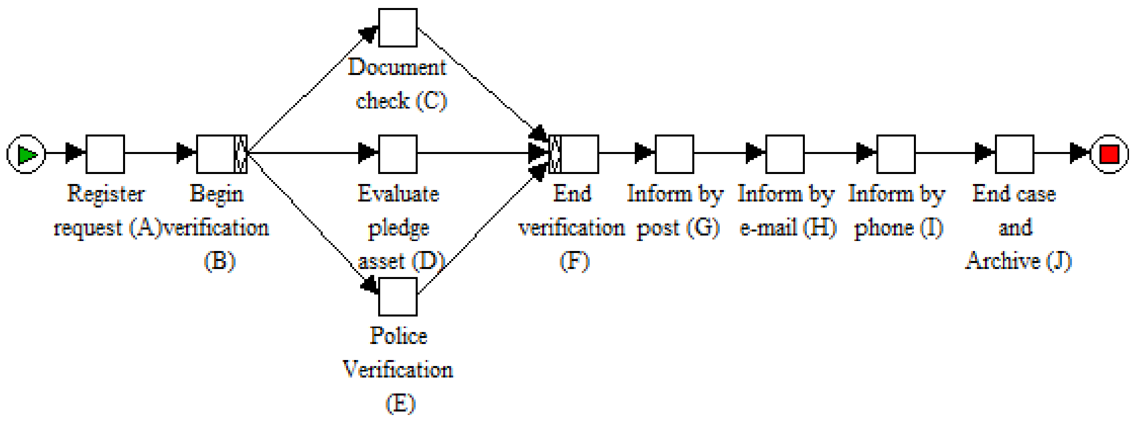


(a) Process model.

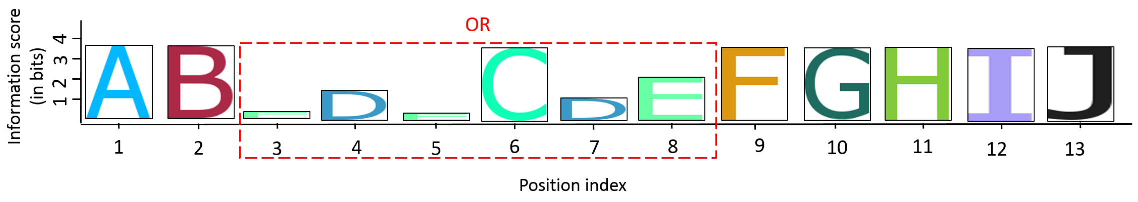


(b) Process logo.

Figure 5.6: Process logo exhibiting AND and XOR split/join.



(a) Process model.



(b) Process logo.

Figure 5.7: Process logo exhibiting OR split/join.

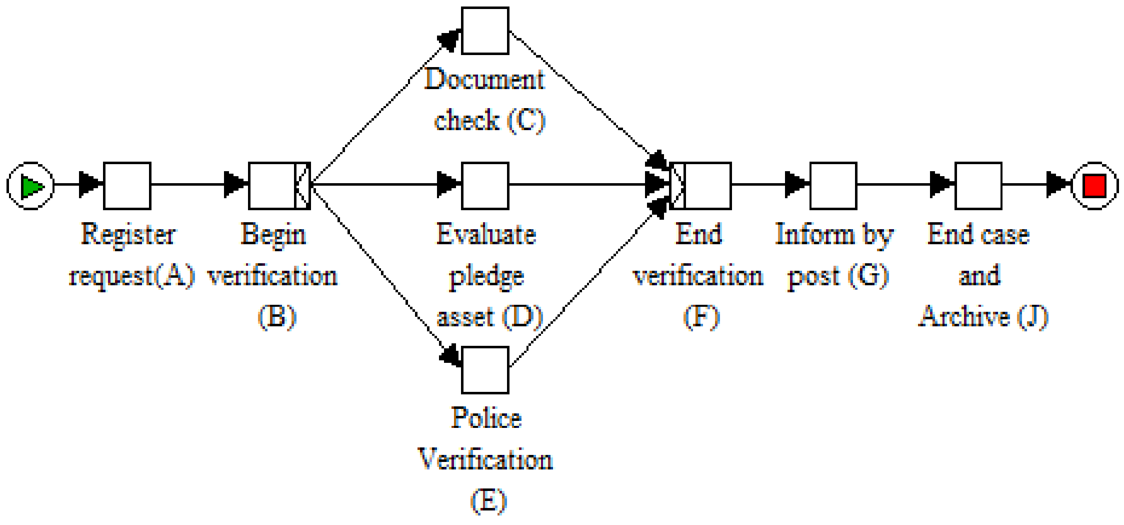
activities where some of the activities can be skipped during the execution period), and three activities are organized between XOR split/join. Activities in process logo corresponding to knock out and XOR are highlighted.

Next variant of loan application handling process shown in fig. 5.6. This process model (shown in fig. 5.6a) consists of AND split/join and XOR split/join at different places. Corresponding process logo is shown in fig. 5.6.

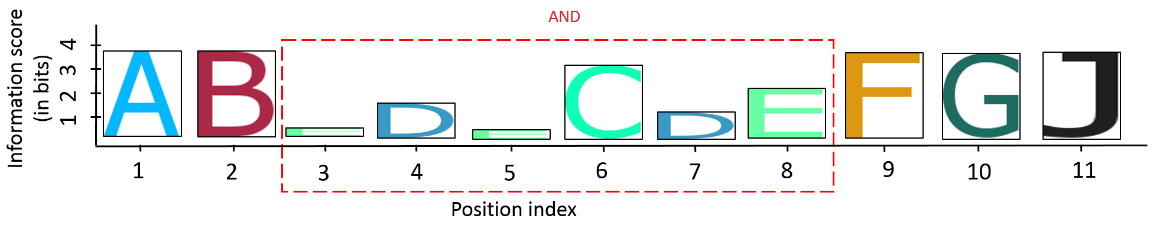
Process variant shown in fig. 5.7a consists of OR split/join. The the corresponding process logo is shown in fig. 5.7b. Slightly different process variant than the one shown in fig. 5.7a is given in fig. 5.8a. The only change is that OR split/join has been replaced by AND split/join and the number of activities are less compared to former one. The process logo corresponding to process variant fig. 5.8a is shown in fig. 5.8b.

Process logo becomes incomprehensible and complex if the process consisting of a large number of activities. To overcome this, traces of the process are clustered, and each cluster can be aligned individually to generate MTA. We demonstrate this with the process model shown in fig. 5.9. It consists of 19 distinct activities.

Traces related to the variant of loan application handling shown in fig.5.9 are



(a) Process model.



(b) Process logo.

Figure 5.8: Process logo exhibiting AND split/join.

clustered to form concrete set of clusters. Each cluster is aligned to generate individual process logo representing the control flow of particular cluster. Process logo shown in fig. 5.10a shows the XOR connections between activities `I` and `J`. Process logo given in fig. 5.10b shows AND split/join between activities `F`, `G` and `H`. Process logo shown in fig. 5.10c shows activity arrangement OR split/join with respect to activities `K`, `L` and `M`. Collectively process logo shown in fig. 5.10a, 5.10b and 5.10c summarizes the control flow of process in fig. 5.9.

5.5 Summary

Relationship between activities constitute the backbone of any process and it models the control flow perspective. A new control flow modeling notation named process logo presented in this chapter is a step towards overcoming limitations in the currently existing control flow modeling notations.

Process logo is capable of efficiently visualizing 1) activity relationships 2) order of

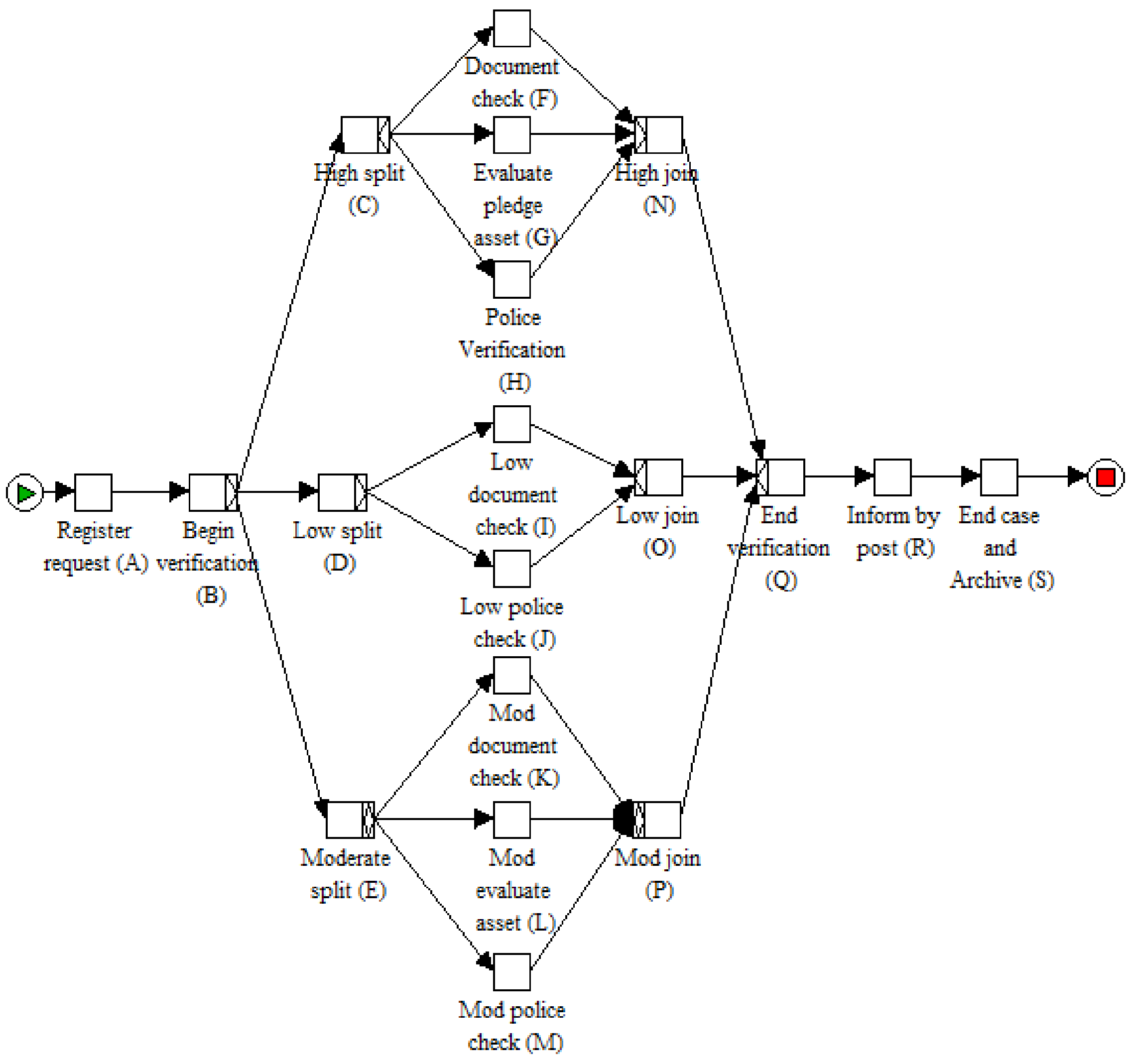
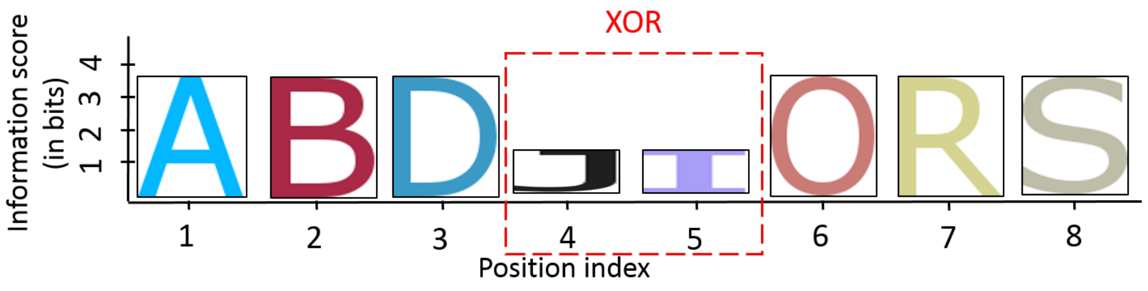
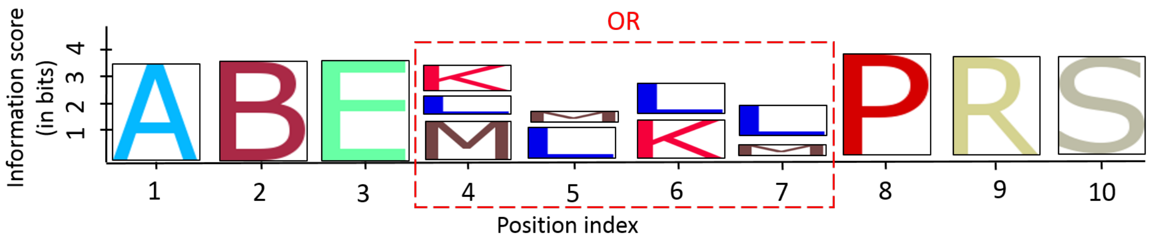


Figure 5.9: Process model with multiple split/join.

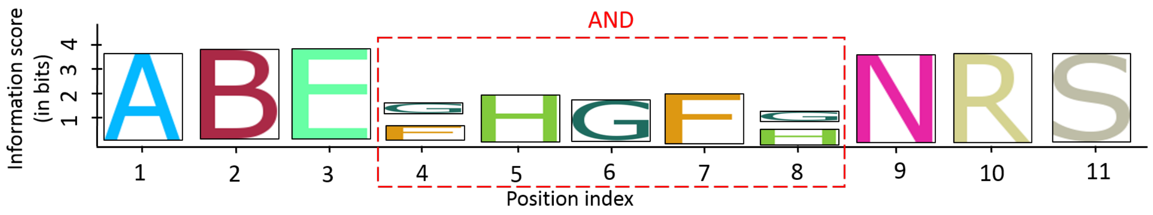
occurrence of activities 3) information score 4) significant activities in a single compact graphic. Process logo is capable of visualizing most commonly known split/join types such as AND, OR, XOR and serial patterns.



(a) Process logo of XOR trace cluster.



(b) Process logo corresponding to OR trace cluster.



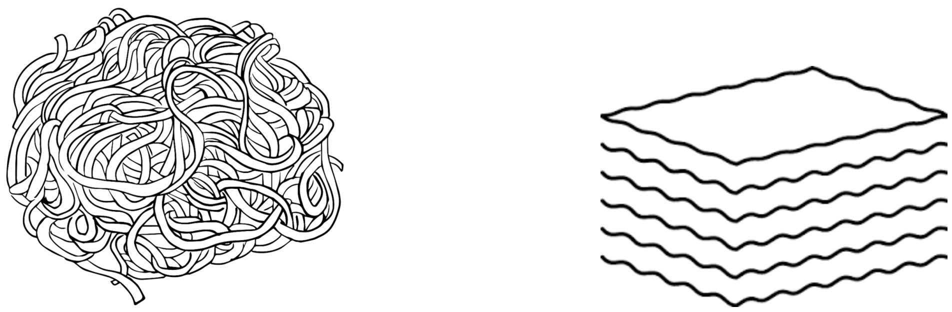
(c) Process logo corresponding to AND trace cluster.

Figure 5.10: Process logo representing clusters of traces shown in figure 5.9.

Chapter 6

Distilling Lasagna from Spaghetti Processes

Functional areas are the fundamental constituent parts of any operational processes. Functional areas facilitate to coordinate, monitor, and execute the processes in an organization. Most typically observed functional areas are production, finance (and accounting), procurement, logistics, resource management, sales, etc. Depending on the structure and complexity of processes in functional areas, it's control flow can be labeled as structured, semi-structured and unstructured.



(a) Spaghetti structure. (b) Lasagna structure.

Figure 6.1: A real-world analogy for Spaghetti and Lasagna processes.

The spectrum of process complexity is continuous. "Unstructured", "semi-structured", and "structured" are used for referring to the same spectrum of process complexity.

- A *Structured process* has a well-defined input and output conditions and matches with observed and expected behavior. These categories of processes can be easily automated.

- In a *semi-structured process*, information requirement of activities are known and it is possible to plan the execution steps much ahead of time.

- In an *unstructured process*, it is highly impossible to define the prior and posterior conditions for executing activities. The fundamental reason is that these processes are carried out by vague qualitative information, intuition, trial-and-error, experience and rules-of-thumb

Any operational process with unstructured control flow is referred as Spaghetti process, and the one with structured control flow is referred as Lasagna process. The term Spaghetti (used for referring unstructured process, an analogy of same is shown in fig. 6.1a) and Lasagna (used for referring structured process, a real life analogy is shown in fig. 6.1b) are adopted in this context for the better understanding of the concept.

Spaghetti and Lasagna control flows are the exact opposite of each other. Spaghetti process is unstructured, cannot be readable, and incomprehensible. Spaghetti processes consist of a large number of activities and transitions. On the other hand, Lasagna processes are highly structured, readable and easily understandable with the manageable number of activities and transitions. Lasagna process is defined with the clear structure and carried out in a predefined manner. The process model is shown in fig. 6.9 is a example for Lasagna process (Process mining tool named “Disco” (Günther and Anne Rozinat, 2012) is used for creating control-flow models used in this chapter) chapter)

Due to the structural complexity of Spaghetti process, only a minuscule of process mining techniques can be used. However, Spaghetti processes are highly interesting from the perspective of process mining, they always allow for continuous improvements. The process model is shown in the fig. 6.2 is a example for Spaghetti process.

The focus of this chapter is to propose techniques for,

- Reducing the structural complexity of the process to extract Lasagna from Spaghetti process.

- Finding the possible and impossible paths of executions in simplified Lasagna process.

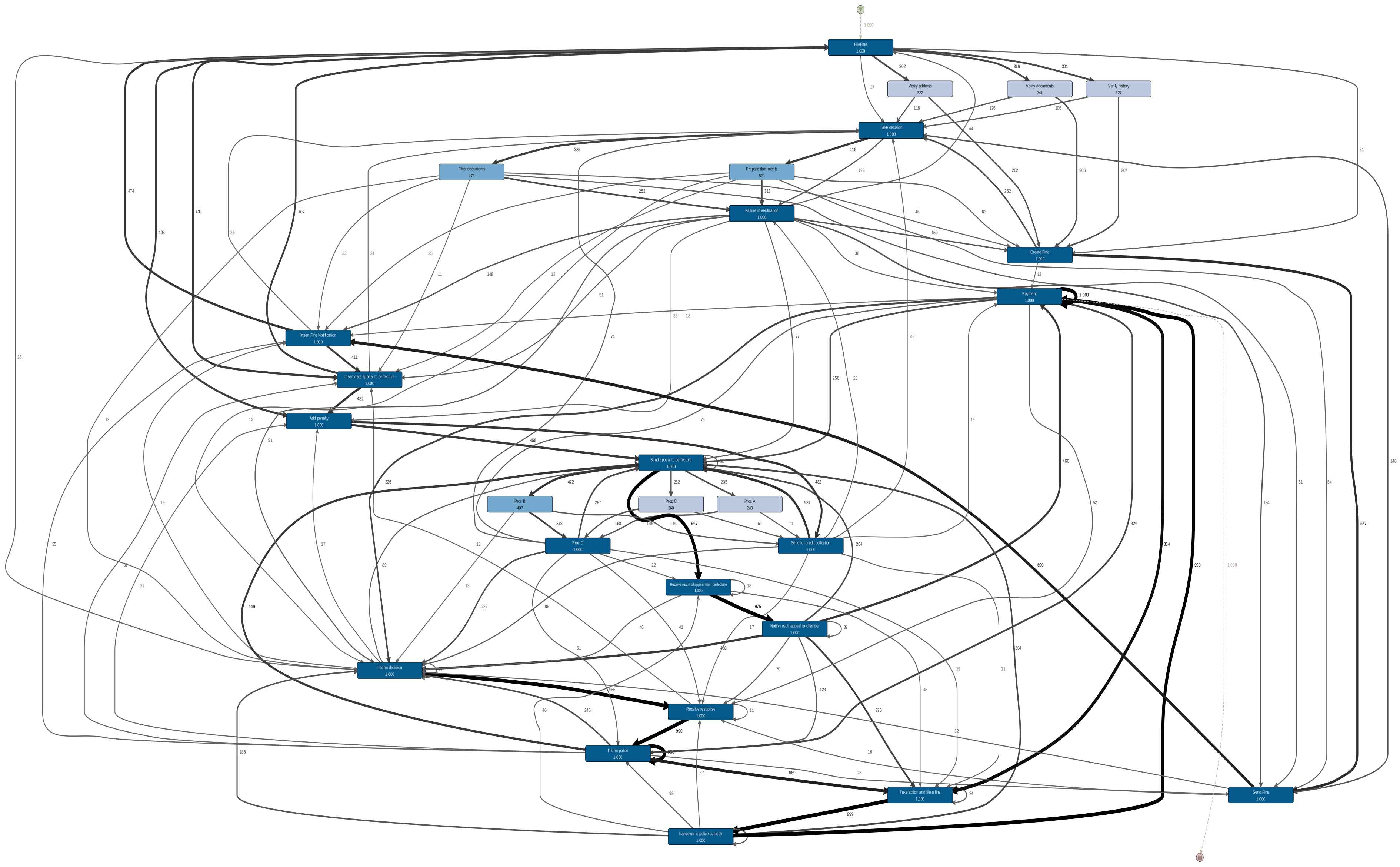

FileFine 1,000
Verify address 332
Verify documents 341
Verify history 327
Take decision 1,000
Filter documents 479
Prepare documents 521
Failure in verification 1,000
Create Fine 1,000
Payment 1,000
Insert Fine Notification 1,000
Insert data appeal to perfecture 1,000
Add penalty 1,000
Send appeal to perfecture 1,000
Proc B 497
Proc C 260
Proc A 243
Proc D 1,000
Send for credit collection 1,000
Receive result of appeal from perfecture 1,000
Notify result appeal to offender 1,000
Inform decision 1,000
Receive response 1,000
Inform police 1,000
Take action and file a fine 1,000
Send Fine 1,000
Handover to police custody 1,000

Figure 6.2: Spaghetti process of road traffic fine management process.

Upcoming contents of this chapter are structured as follows. Typical functional areas are explained in section 6.1. Section 6.2 presents the framework. The method for converting Spaghetti to Lasagna is explained in section 6.3. The technique to find possible and impossible paths of execution is discussed in section 6.4. The result of experimental work is documented in section 6.5. Section 6.6 presents the upshot of the chapter.

6.1 Common Functional Areas in Organization

The structure of process (Lasagna or Spaghetti) associated with various functional areas of an organization is necessary for process management. Most commonly observed functional areas in an organization are shown in fig. 6.3.

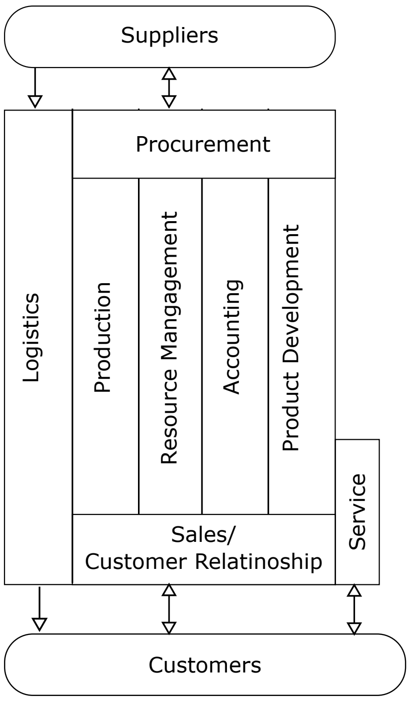


Figure 6.3: Overview of the different functional areas in a typical organization.

Both types of processes can be found in all of the functional areas. Lasagna processes are typically encountered in production, finance/accounting, procurement, logistics, resource management, and sales/CRM. Spaghetti processes are typically encountered in product development, service, resource management, and sales/CRM.

6.2 Framework

The framework used for extracting Lasagna from Spaghetti process is shown in fig. 6.4. Initially, traces in the event log are read and control flow model is constructed. Constructed model is checked against Lasagna properties. If the control flow structure does not match with the Lasagna properties, then the activities and transitions are aggregated and abstracted with the help of fuzzy miner (Günther and Van Der Aalst, 2007). This step is repeated until the formation of a concrete Lasagna control flow model.

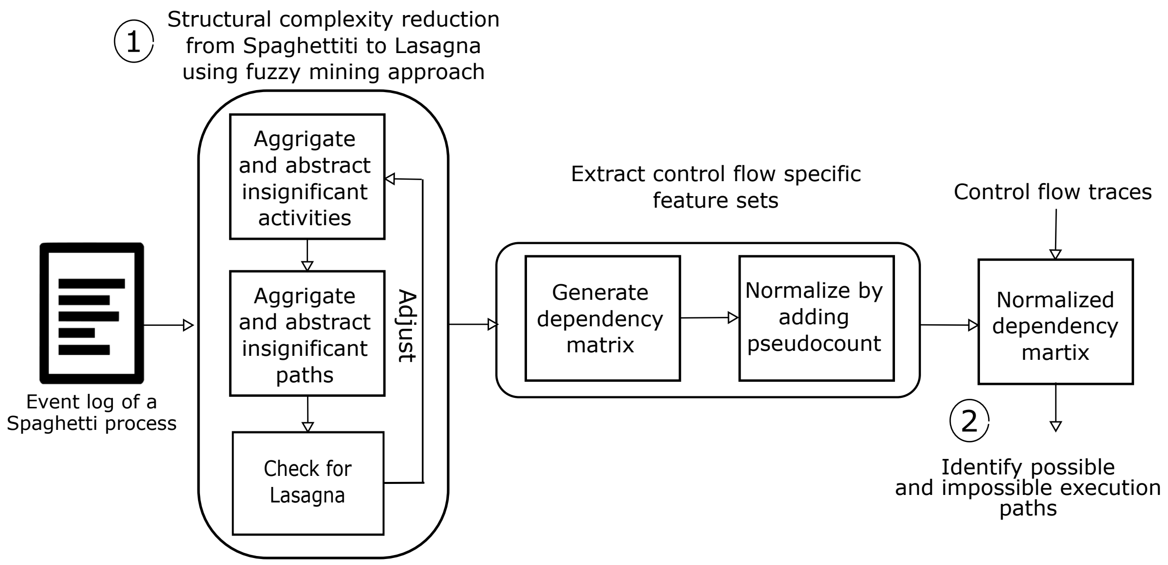

1
Structural complexity reduction from Spaghettiti to Lasagna using fuzzy mining approach
Aggrigate and abstract insignificant activities
Aggrigate and abstract insignificant paths
Check for Lasagna
Adjust
Event log of a Spaghetti process
Extract control flow specific feature sets
Generate dependency matrix
Normalize by adding pseudocount
Control flow traces
Normalized dependency martix
2
Identify possible and impossible execution paths

Figure 6.4: Framework for simplifying Spaghetti processes.

The validity of the simplified model is verified based on its fitness value (Van Der Aalst, 2011). If the fitness value (discovered model should allow for the behavior seen in the event log. A model having a good fitness can replay most of the traces in the log) of the simplified model is ≥ 0.8, then the process is considered as final Lasagna process (will not be simplified any further). Once the Lasagna process model is extracted, control flow specific features (such as absolute frequency and dependency matrices) are calculated. These feature values are used in finding the possible and impossible traces in the process.

6.3 From Spaghetti to Lasagna

We use the following definition (rule of thumb) of Lasagna process stated by Van Der Aalst (2011) for validating the Lasagna property of simplified process.

"*A process is a Lasagna process if with limited efforts it is possible to create an agreed-upon process model that has a fitness of at least* 0.8, *i.e., more than* 80% *of the events happen as planned*".

Which means, more than 80% of the events happen as planned and stakeholders confirm the validity of the model.

6.4 Finding Possible and Impossible Traces

We obtain absolute frequency, and it is utilized to discover the corresponding dependency matrix. Scores in dependency matrix are used for determining the possible or impossible traces.

6.4.1 Absolute Frequency Matrix (AFM)

Connection between activities can be represented using follows and/or precedes relationships. For any given pair of activities $\mathtt{a}$ and $\mathtt{b}$ we can determine whether they sometimes, always, or never *follows* and/or *precedes* each other. The follows or precedes relationships are sufficient enough to discover how significantly the activities are interlinked.

AFM is constructed using follows relationship. It depicts the number of times one activity is directly followed by another activity. Values in AFM is calculated using equation 6.1.

Definition 6.1. (Absolute follows relationship): Let $\mathcal{L}$ be the event log over $\mathcal{A}$ and $\mathtt{a}, \mathtt{b} \in \mathcal{A}$. $|\mathtt{a} >_{\mathcal{L}} \mathtt{b}|$ is the number of times $\mathtt{a}$ is directly followed by $\mathtt{b}$ in $\mathcal{L}$, i.e.,

$$|\mathtt{a} >_{\mathcal{L}} \mathtt{b}| = \Sigma_{\sigma \in \mathcal{L}} \mathcal{L}(\sigma) \times |\{1 \leq i \leq |\sigma| | \sigma(i) = \mathtt{a} \wedge \sigma(i+1) = \mathtt{b}\}| \tag{6.1}$$

6.4.2 Dependency Matrix

Values of AFM is used for calculating dependency matrix. Dependency matrix signifies the strength of the relationship between activities of a process. Dependency matrix is constructed by equation 6.2.

Definition 6.2. (Dependency value): $|a \Rightarrow b|$ is the value of the dependency relation between $\mathtt{a}$ and $\mathtt{b}$,

$$|\mathtt{a} \Rightarrow \mathtt{b}| = \begin{cases} \frac{|\mathtt{a} >_{\mathcal{L}} \mathtt{b}| - |\mathtt{b} >_{\mathcal{L}} \mathtt{a}|}{|\mathtt{a} >_{\mathcal{L}} \mathtt{b}||\mathtt{b} >_{\mathcal{L}} \mathtt{a}| + 1} & \text{if } \mathtt{a} \neq \mathtt{b} \\ \frac{\mathtt{a} >_{\mathcal{L}} \mathtt{a}|}{|\mathtt{a} >_{\mathcal{L}} \mathtt{a}| + 1} & \text{if } \mathtt{a} = \mathtt{b} \end{cases} \tag{6.2}$$

Dependency value varies between 1 and -1.

- Two activities $\mathtt{a}$ and $\mathtt{b}$ are highly dependent if the value of $|\mathtt{a} \Rightarrow_{\mathcal{L}} \mathtt{b}|$ is close to 1.
- If the value of $|\mathtt{a} \Rightarrow_{\mathcal{L}} \mathtt{b}| \approxeq -1$ then there is a high negative dependency between $\mathtt{a}$ and $\mathtt{b}$.

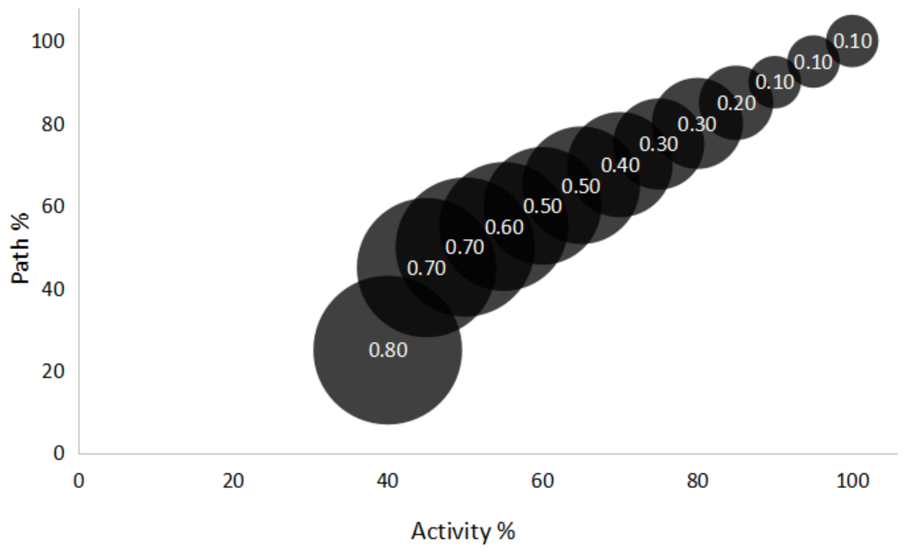


Figure 6.5: Control-flow fitness versus activity and path percentage.

6.5 Results

The methods for simplifying spaghetti processes to find the possible and impossible paths of execution is evaluated on event log of *road traffic fine management process* (control flow model of this process is shown in fig. 6.2) (De Leoni and Mannhardt, 2015). It is taken from the standard process mining event log repository (4TU.Centre for Research Data). This event log made of 561, 470 events divided over 150, 370 cases and consists of 27 distinct activities.

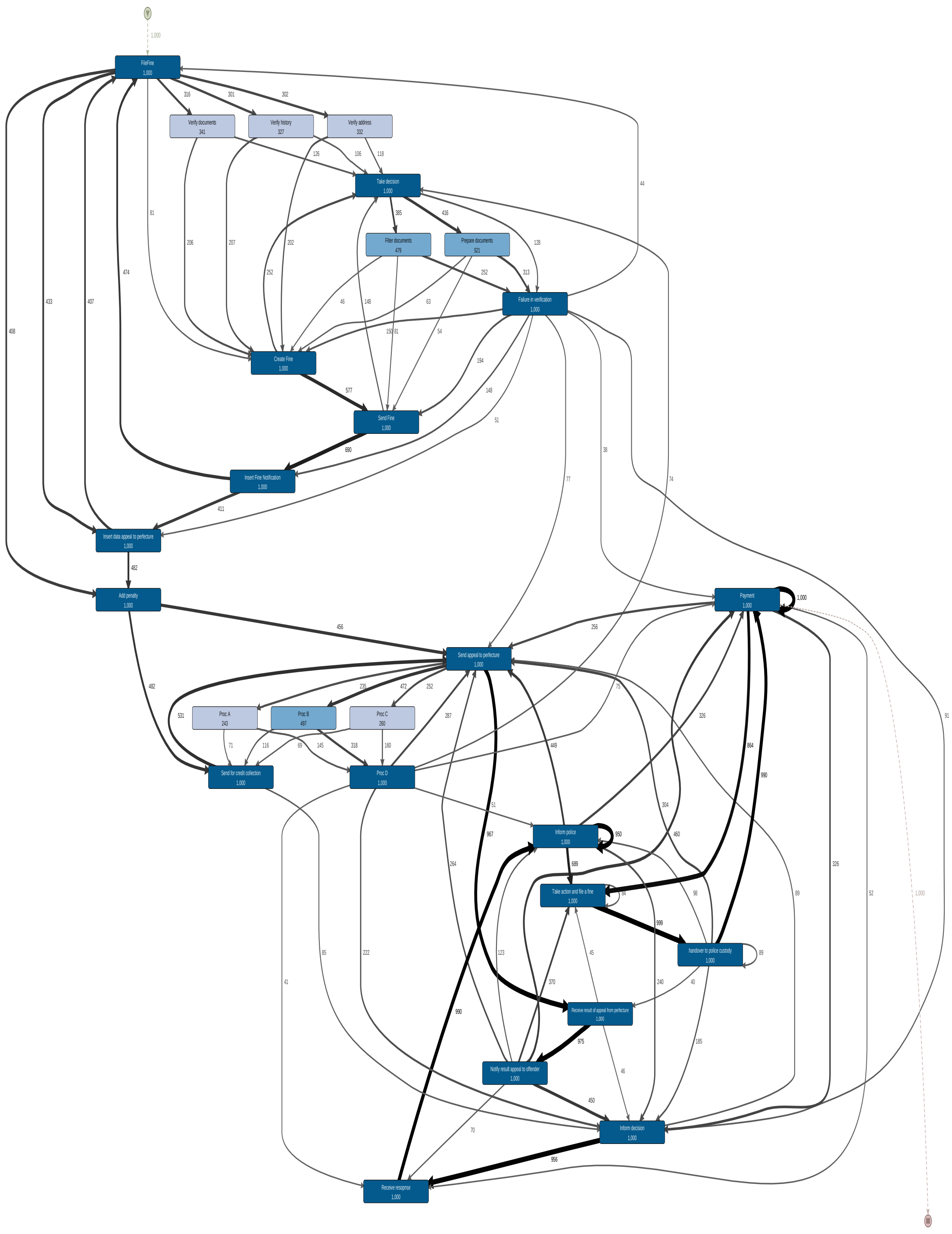
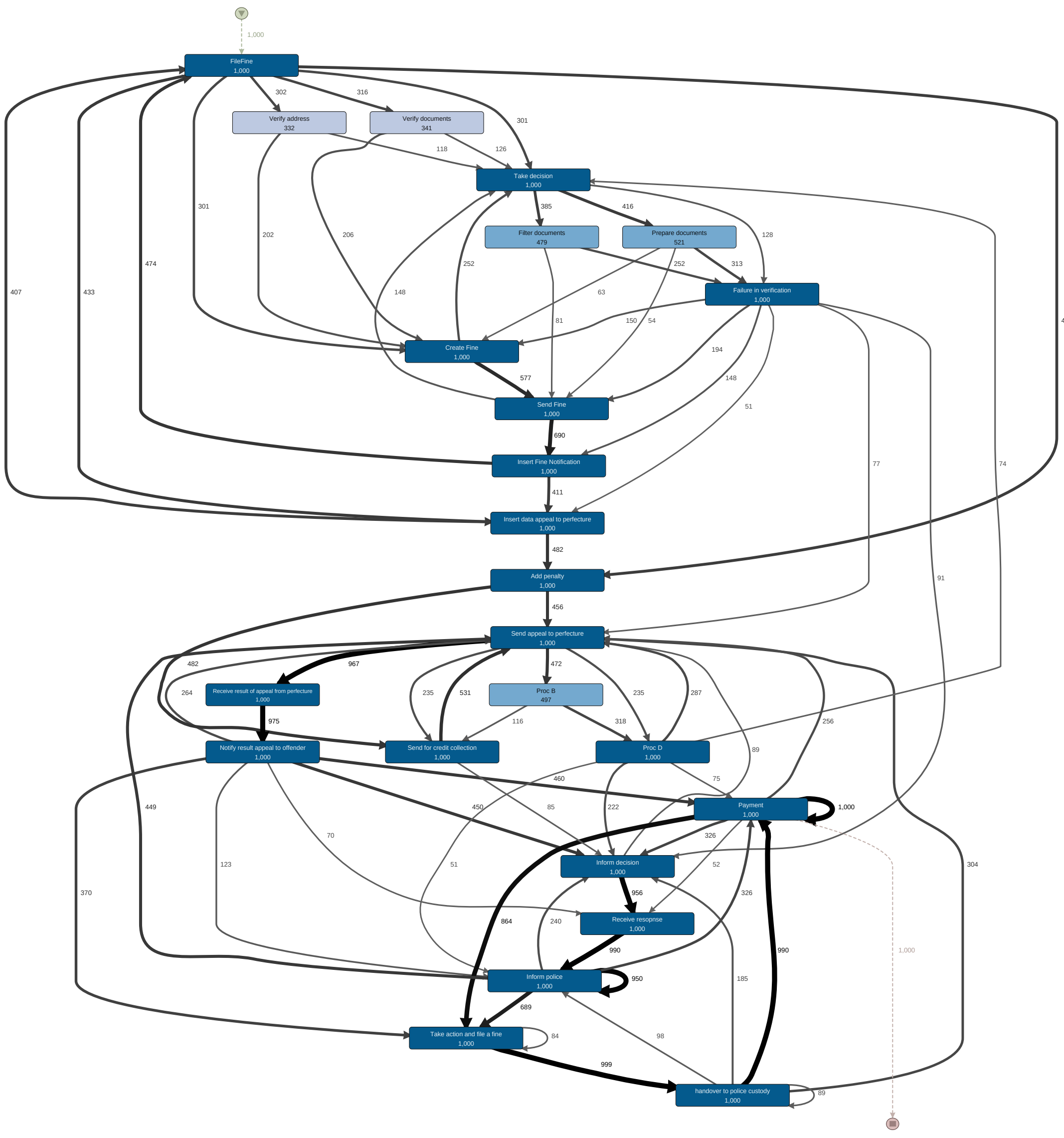


Figure 6.6: Road traffic process with activities: 80%, Paths: 40%, Fitness:0.3.

6.5.1 Step 1: Process simplification

Fuzzy miner (Günther and Van Der Aalst, 2007) is employed on initial spaghetti process (fig. 6.2), and the control flow of road traffic fine management process is



1,000
FileFine 1,000
Verify address 332
Verify documents 341
Take decision 1,000
Filter documents 479
Prepare documents 521
Failure in verification 1,000
Create Fine 1,000
Send Fine 1,000
Insert Fine Notification 1,000
Insert data appeal to perfecture 1,000
Add penalty 1,000
Send appeal to perfecture 1,000
Receive result of appeal from perfecture 1,000
Proc B 497
Notify result appeal to offender 1,000
Send for credit collection 1,000
Proc D 1,000
Payment 1,000
Inform decision 1,000
Receive response 1,000
Inform police 1,000
Take action and file a fine 1,000
handover to police custody 1,000

Figure 6.7: Road traffic process with activities: 65%, Paths: 30%, Fitness:0.6.

constructed at various levels of activity and path aggregation and abstraction. Fitness value of generated control flow models are computed and checked against Lasagna definition, this process is repeated iteratively until finding a process model with the fitness value ≥ 0.8.

The intermediate control flow models of road traffic fine management process with various levels of activity and path abstractions are shown in fig. 6.6 - 6.8. Considerable improvement in fitness value can be seen by reducing the complexity of the process. Balloon graph is shown in fig. 6.5 illustrates the effect of decreasing process complexity (in terms of activity and paths) on fitness value. It is observed that complexity and fitness are inversely proportional.

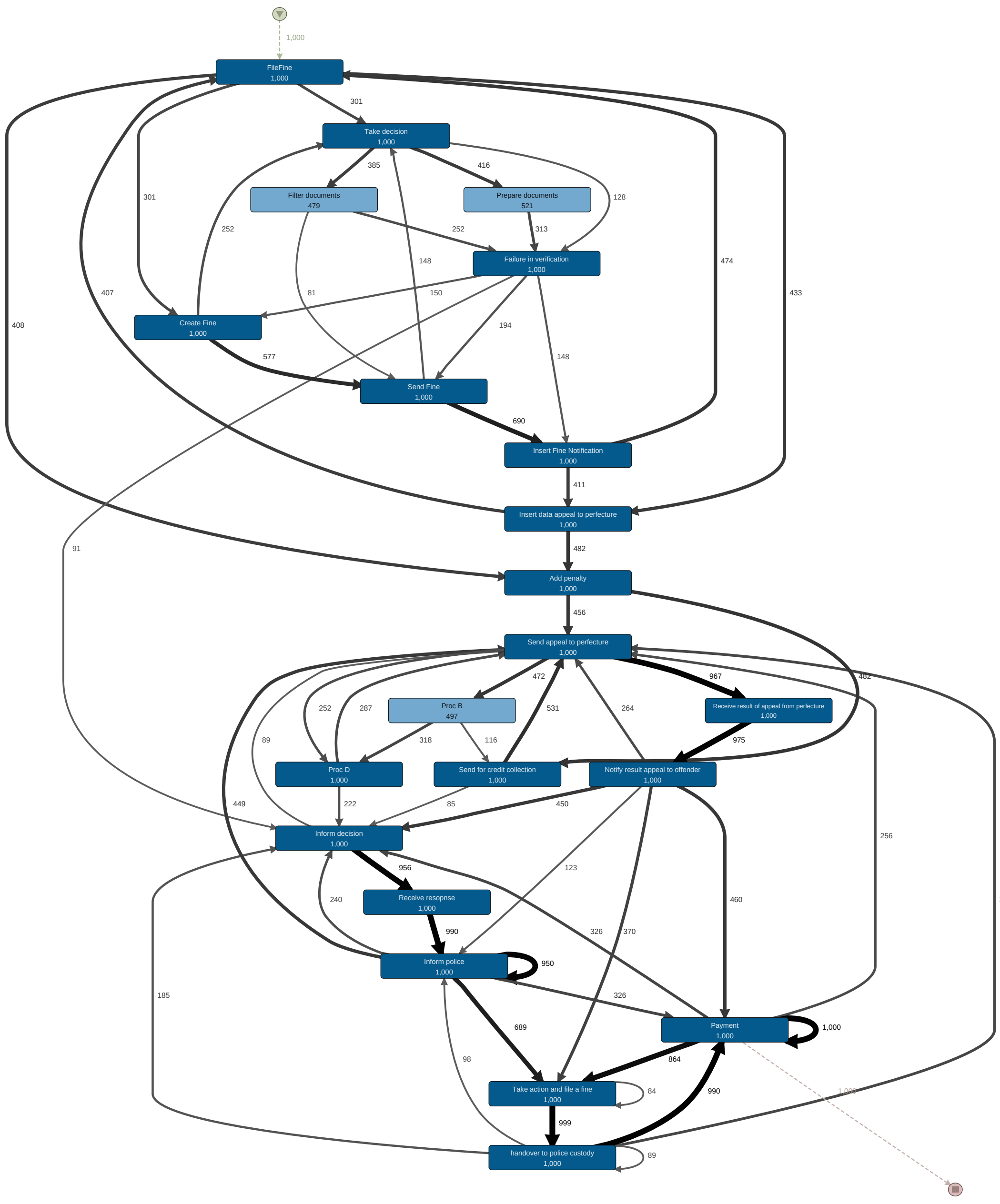

FileFine 1,000
Take decision 1,000
Filter documents 479
Prepare documents 521
Failure in verification 1,000
Create Fine 1,000
Send Fine 1,000
Insert Fine Notification 1,000
Insert data appeal to perfecture 1,000
Add penalty 1,000
Send appeal to perfecture 1,000
Proc B 497
Receive result of appeal from perfecture 1,000
Proc D 1,000
Send for credit collection 1,000
Notify result appeal to offender 1,000
Inform decision 1,000
Receive resopnse 1,000
Inform police 1,000
Payment 1,000
Take action and file a fine 1,000
handover to police custody 1,000

Figure 6.8: Road traffic process with activities: 50%, Paths: 25%, Fitness:0.7.

The simplified Lasagna model of road traffic fine management process is shown in fig. 6.9. It has fitness value of 0.8 and it is constructed with 40% of activity and 25% of path abstraction and aggregation. The simplified Lasagna process is used to extract the control flow features to assist us in determining the set of possible and impossible traces.

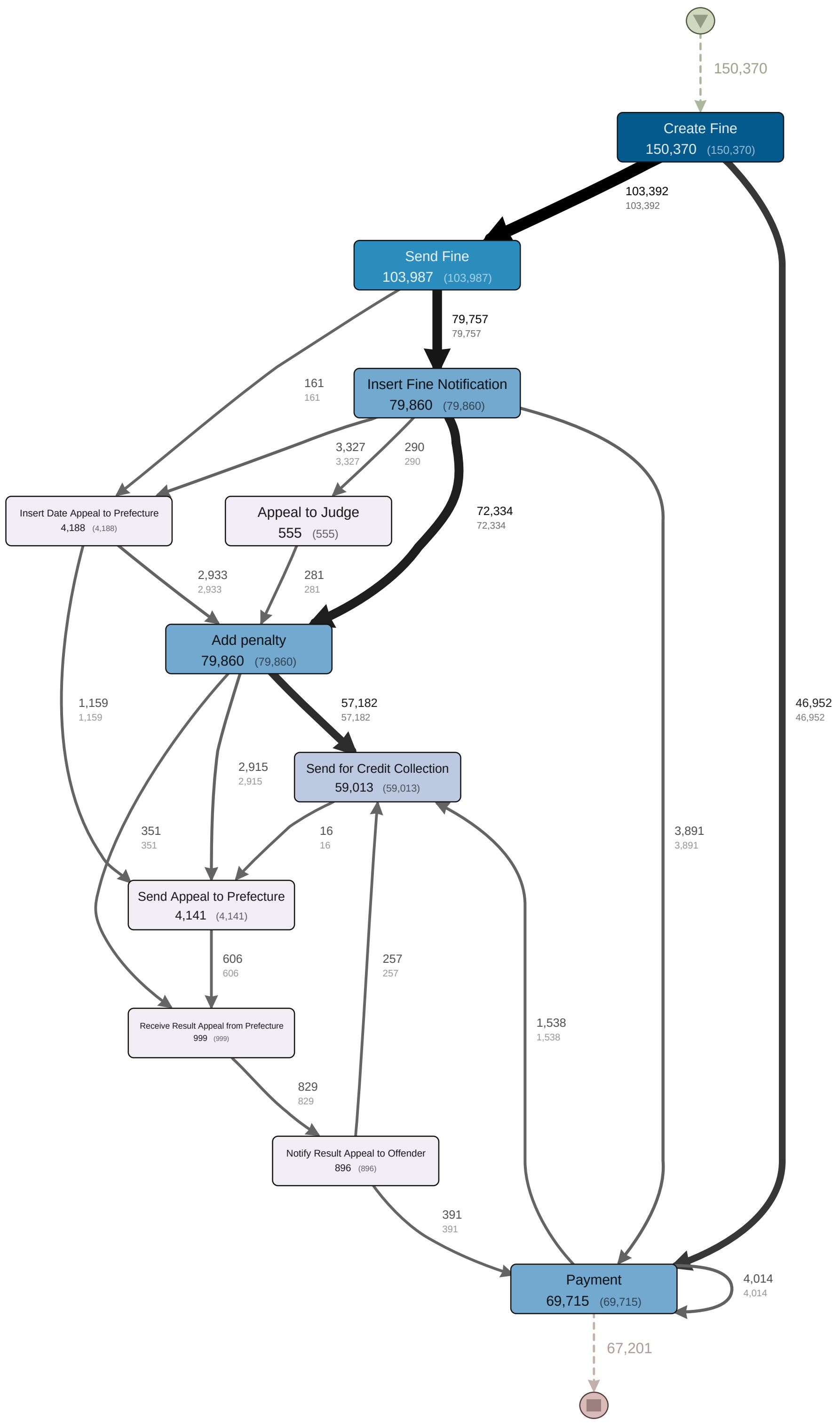


Figure 6.9: Lasagna model of road traffic fine management process.

Table 6.1: Absolute Frequency Matrix.

$\| >_{\mathcal{L}} \|$	a	b	c	d	e	f	g	h	i	j	k
a	0	103392	0	0	0	0	0	0	0	0	46952
b	0	0	75757	161	0	0	0	0	0	0	0
c	0	0	0	3327	290	72334	0	0	0	0	3891
d	0	0	0	0	0	2933	0	1159	0	0	0
e	0	0	0	0	0	281	0	0	0	0	0
f	0	0	0	0	0	0	57182	2915	351	0	0
g	0	0	0	0	0	0	0	16	0	0	0
h	0	0	0	0	0	0	0	0	606	0	0
i	0	0	0	0	0	0	0	0	0	829	0
j	0	0	0	0	0	0	257	0	0	0	391
k	0	0	0	0	0	0	1538	0	0	0	4014

Table 6.2: Dependency matrix.

$\Rightarrow_{\mathcal{L}}$	a	b	c	d	e	f	g	h	i	j	k
a	0	0.99	0	0	0	0	0	0	0	0	0.99
b	-0.49	0	0.99	0.99	0	0	0	0	0	0	0
c	0	-0.49	0	0.99	0.99	0.99	0	0	0	0	0.99
d	0	-0.49	-0.49	0	0	0.99	0	0.99	0	0	0
e	0	0	-0.49	0	0	0.99	0	0	0	0	0
f	0	0	-0.49	-0.49	-0.49	0	0.99	0.99	0.99	0	0
g	0	0	0	0	0	-0.49	0	0.94	0	-0.99	-0.99
h	0	0	0	-0.49	0	-0.49	-0.48	0	0.99	0	0
i	0	0	0	0	0	-0.49	0	-0.49	0	0.99	0
j	0	0	0	0	0	0	1.00	0	-0.49	0	0.99
k	-0.49	0	-0.49	0	0	0	1.00	0	0	-0.49	0.99

6.5.2 Step 2: Finding the possible and impossible paths of execution

For easily referring the activities in Lasagna process, we assign simple one letter names for each of the 11 activities shown in fig. 6.9. Short names are, file fine (a), create fine (`b`), send fine (`c`), insert fine notification (`d`), insert data appeal to perfecture (`e`), add penalty (`f`), send for credit collection (`g`), send appeal to perfecture (`h`), receive the result of appeal from perfecture (i), Notify the result of appeal to offender (`j`), and payment (`k`).

A AFM and dependency matrix

AFM of road traffic fine management process is given in table 6.1. For instance, $|\mathtt{b} >_{\mathcal{L}} \mathtt{d}| = 161$, i.e., in the entire log, activity named `d` has been carried out 161 times after the end of activity `b`. Absolute frequency matrix related to simplified control flow of road traffic fine management process is given in table 6.1.

Dependency matrix corresponding to simplified Lasagna process (fig. 6.9) is shown in table 6.2. The values in this matrix will be used for distinguishing between possible and impossible sets of traces.

B Possible and impossible paths

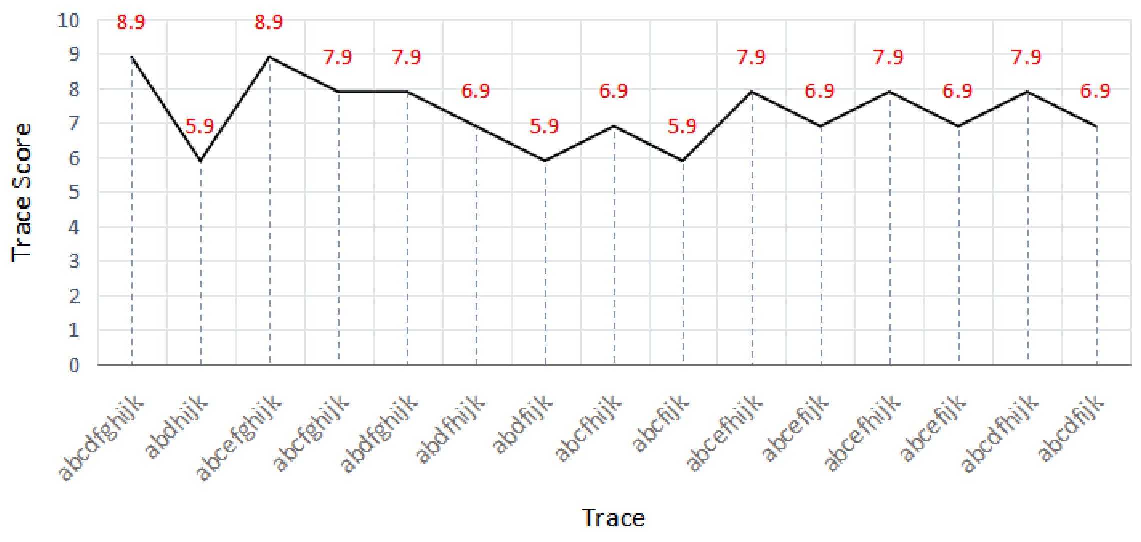
Set of random traces are constructed using the activities in simplified Lasagna process (fig. 6.9). Value for each trace is computed by adding corresponding values of dependency matrix given in table 6.2. The premise is, traces that are according to control-flow of Lasagna process obtain considerably high score compared to the traces that are not confirming to control-flow of Lasagna process.

Following are the sets containing possible and impossible traces we use for evaluating proposed techniques.

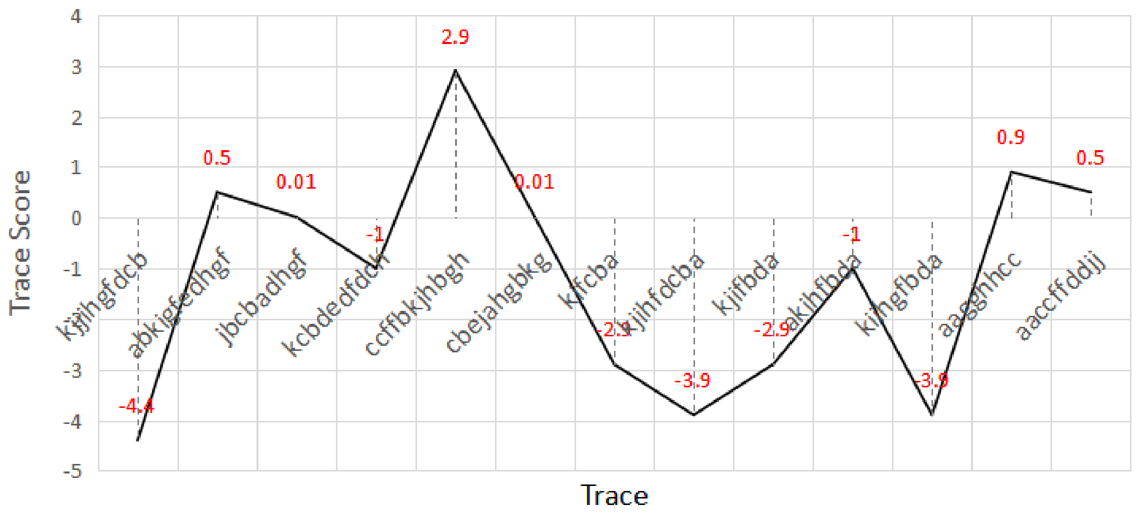
$T_p = \{$ `abcdfghijk, abdhijk, abcefghijk, abcfghijk, abdfghijk, abdfhijk, abdfijk, abcfhijk, abcfijk, abcefhijk, abcefijk, abcefhijk, abcefijk, abcdfhijk, abcdfijk`$\}$

$T_i = \{$`kjjihgfdcb, abkigfedhgf, jbcbadhgf, kcbdedfdch, ccffbkjhbgh, cbejahgbkg, kifcba, kjihfdcba, kjifbda, akihfbda, kiihgfbda, aagghhcc, aaccffddjj`$\}$

Traces in the set T_p are formed according to the control flow structure shown in the fig. 6.9. Traces in the set T_i are impossible to be run on control flow shown in fig.



(a) Scores for possible paths of execution.



(b) Scores for impossible paths of execution.

Figure 6.10: Possible and impossible paths of execution.

6.9. Trace value for each trace in both sets is calculated using activity dependency values given in table 6.2.

The result of evaluating the set of possible and impossible traces is shown in fig. 6.10. The graph is shown in fig. 6.10a corresponds to possible set of traces (T_p) and graph shown in fig. 6.10b corresponds to impossible set of traces (T_i). From the graphs shown in fig. 6.10, scores obtained by traces in T_p is always positive and relatively higher to the scores of T_i. This result validates that proposed method is capable of identifying the set of possible and impossible traces in any Spaghetti structured process.

6.6 Summary

Spaghetti processes are unstructured, difficult to understand and impossible to comprehend. Due to its structural complexity, most of the currently available techniques in process mining can't be applied. Methods and techniques for analysis of spaghetti process are always a matter of interest in the process mining research community.

This chapter demonstrated the method for extracting Lasagna (structured) process from Spaghetti (unstructured) process. Further, the method has been proposed for identifying the set of possible and impossible traces on the simplified process. The proposed methods have been applied and validated on real world event log.

Chapter 7

Identifying Frequent Execution Paths in Information System

Availability of event log and process model play a significant role while introducing new information systems. Formal and informal are the two categories of models used for designing and redesigning information systems.

- *Informal model* is used for discussion and documentation (typically ambiguous and vague).
- *Formal model* (also referred to as executable model) is used for analyzing and enacting the actual execution of a process (these models tend to have the specific focus and too detailed, difficult to be understandable by the stakeholders).

The lack of alignment between formal and informal models has been discussed extensively in BPM literature (Dumas et al., 2005; Harel and Marelly, 2003; Ter Hofstede et al., 2009).

The value of the model is limited if too little attention is paid to the alignment of model and reality. Process models become “paper tigers” when the people involved cannot trust them. For example, it makes no sense to conduct simulation experiments while using a model that assumes an idealized version of the real process. It may lead to the incorrect design and redesign decisions. It is also difficult to start a new implementation project guided by process models that hide the reality.

A process model used to configure a workflow management system is probably well-aligned with reality. Unfortunately, most hand-made models are disconnected from reality and provide only an idealized view of the processes at hand.

Table 7.1: Event-log of online shopping process.

Case Id	Event Id	Activity	Resource	Timestamp		Cost	Data
				Start Time	End Time		
...	...	...	...	...	...	...	...
	2342	Receive order	Anne	10-10-2012/1:20	10-10-2012/1:24	10	Order particulars
	2343	Verify contact	Jhon	10-10-2012/1:24	10-10-2012/1:29	20	Contact information
	2344	Verify cust. history	Thomas	10-10-2012/1:24	10-10-2012/1:29	5	History of purchases by the same customer
	2345	Verify order	Clar	10-10-2012/1:24	10-10-2012/1:25	5	Order information, address, and quantity
	2346	Decide	Ram	10-10-2012/1:25	10-10-2012/1:27	50	Relevant information specific to order
xx12	2347	Confirm order	Pete	10-10-2012/1:29	10-10-2012/1:34	5	Order information and reason
	2348	Inform by email	Wil	10-10-2012/1:34	10-10-2012/1:35	2	Message
	2349	Prepare shipment and invoice	Jeni	10-10-2012/1:34	10-10-2012/1:36	20	Order information
	2350	Deliver order	Susan	10-10-2012/1:37	10-10-2012/1:38	20	Delivery address and amount
	2351	Receive payment	Peter	10-10-2012/1:37	10-10-2012/1:38	35	Order number
	2352	Close order	Peter	10-10-2012/1:37	10-10-2012/1:38	0	Order number
	7655	Receive order	Anne	7-11-2012/7:10	7-11-2012/7:15	5	Order particulars
	7656	Verify contact	John	7-11-2012/7:15	7-11-2012/7:17	20	Contact information
	7657	Verify cust. history	Thomas	7-11-2012/7:15	7-11-2012/7:20	10	History of purchases by the same customer
xx13	7658	Verify order	Clar	7-11-2012/7:17	7-11-2012/7:20	10	Order information, address, and quantity
	7659	Decide	Ram	10-11-2012/7:20	11-11-2012/7:20	20	Relevant information specific to order
	7657	Reject order	Steven	11-11-2012/7:20	12-11-2012/7:20	10	Reason
	7658	Close order	Peter	12-11-2012/7:20	13-11-2012/7:20	0	Order number
xx14	9875	Receive order	Anne	24-11-2013/9:30	24-11-2013/10:30	100	Order particulars
...	...	...	...	...	...	...	...

Here the focus is on formal models. in the aspect of formal models, rigorous analysis techniques may have little to do with the actual process. But, these are the models used for actual implementation and analysis.

Knowledge about frequent execution paths in formal models helps stakeholders to discuss, optimize, verify, analyze, animate, specify and configure the process related information. *We propose a method for identifying frequent paths of executions with the help of formal models.* The focus of this chapter is to propose the methods and techniques for,

- Capturing the process execution information.

- Identifying the frequent execution patterns.

Upcoming sections of this chapters are organized follows. Section 7.1 introduces online shopping process which is used as the case study for validating results in this chapter. Framework for identifying frequent execution paths is explained in section 5.2. Extraction of feature set needed for identifying frequent execution path is illustrated in section the 7.3. Results of experimental study is given in section 7.4. Finally, this chapter concludes with a summary.

7.1 Process Model Used for Case Study

Petri-net (Peterson, 1981) representing the control-flow model of online shopping website is shown in the fig. 7.1. It is constructed by applying α algorithm (Van Der Aalst et al., 2004) on the process log given in the table 7.1

According to fig. 7.1, once the order is received (`a`), verification steps (`b`,`c` and `d`) are carried out before approving (`f`) or rejecting (`j`) the order. Order is closed immediately if the decision is to reject the order. Upon approving the order, customer will be notified (`h`, `i`) and the consignment with invoice is sent (`g`). After delivering (`k`) and receiving the payment (`i`), order will be closed (`m`). Table 7.2 shows the set of all traces possible on online shopping process model.

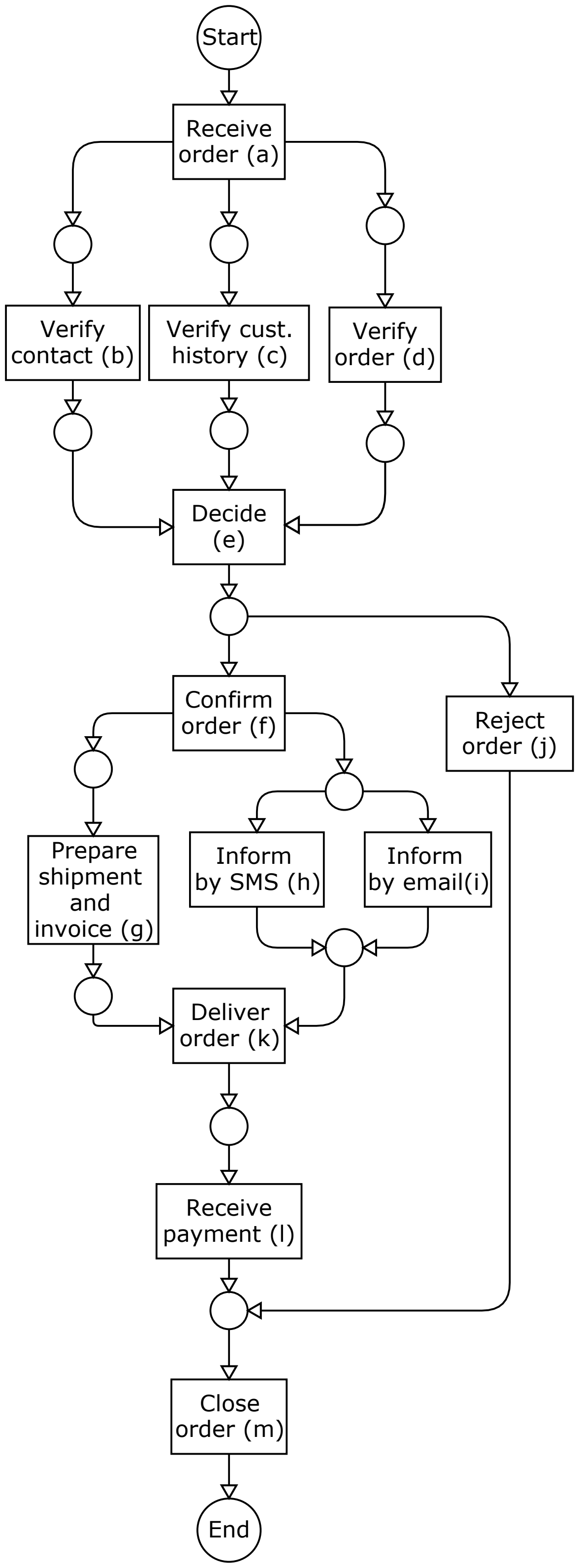


Figure 7.1: Control-flow model of online shopping process.

Table 7.2: Possible traces with positional information.

Activity positions										
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
a	b	c	d	e	f	g	h	k	l	m
	b	d	c			h	g			
	d	c	b			g	i			
	d	b	c			i	g			
	c	b	d			j	m			
	c	d	b							

7.2 Framework

Steps followed for identifying frequent execution patterns is shown in fig. 7.2. Initially, a event log is processed to extract the control-flow traces. Traces are used to generate the alignment as shown in table 7.3. Aligned traces are used to extract Position Specific Scoring Matrix (PSSM) and Position Probability Matrix (PPM). PSSM and PPM serve as the building block for identifying frequent control-flow execution patterns.

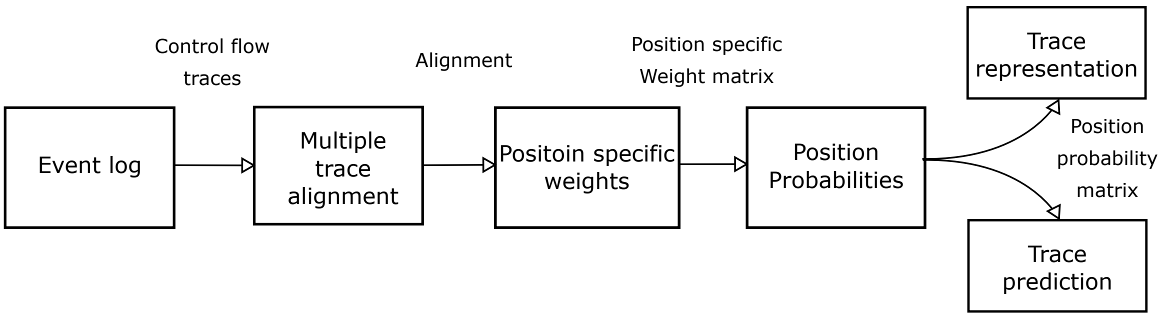


Figure 7.2: Framework for predicting the frequent execution patterns.

7.3 From Traces to Position Weight Matrix

Position Weight Matrix (PWM) is a trace descriptor. It attempts to capture the intrinsic variability characteristic of activities in trace patterns. A Profile is usually derived from a set of functionally related aligned control-flow traces. A PWM is a two-dimensional matrix. It has one row for each activity appearing in the trace alignment and one column for every position in the alignment.

Table 7.3: Aligned traces of online shopping process.

Number	**Control-flow trace**
Trace 1:	a, b, c, d, e, f, g, h, k, l, m
Trace 2:	a, b, d, c, e, f, g, h, k, l, m
Trace 3:	a, d, c, b, e, f, g, h, k, l, m
Trace 4:	a, d, b, c, e, f, g, h, k, l, m
Trace 5:	a, c, b, d, e, f, g, h, k, l, m
Trace 6:	a, c, d, b, e, f, g, h, k, l, m
Trace 7:	a, b, c, d, e, f, h, g, k, l, m
Trace 8:	a, b, d, c, e, f, h, g, k, l, m
Trace 9:	a, d, c, b, e, f, h, g, k, l, m
Trace 10:	a, d, b, c, e, f, h, g, k, l, m
Trace 11:	a, c, b, d, e, f, h, g, k, l, m
Trace 12:	a, c, d, b, e, f, h, g, k, l, m
Trace 13:	a, b, c, d, e, f, g, i, k, l, m
Trace 14:	a, b, d, c, e, f, g, i, k, l, m
Trace 15:	a, d, c, b, e, f, g, i, k, l, m
Trace 16:	a, d, b, c, e, f, g, i, k, l, m
Trace 17:	a, c, b, d, e, f, g, i, k, l, m
Trace 18:	a, c, d, b, e, f, g, i, k, l, m
Trace 19:	a, b, c, d, e, f, i, g, k, l, m
Trace 20:	a, b, d, c, e, f, i, g, k, l, m
Trace 21:	a, d, c, b, e, f, i, g, k, l, m
Trace 22:	a, d, b, c, e, f, i, g, k, l, m
Trace 23:	a, c, b, d, e, f, i, g, k, l, m
Trace 24:	a, c, d, b, e, f, i, g, k, l, m
Trace 25:	a, b, c, d, e, f, j, m
Trace 26:	a, b, d, c, e, f, j, m
Trace 27:	a, b, d, c, e, f, j, m
Trace 28:	a, d, c, b, e, f, j, m
Trace 29:	a, d, b, c, e, f, j, m
Trace 30:	a, c, b, d, e, f, j, m

7.3.1 Absolute Frequency Matrix (AFM)

Generating AFM is the first step towards creating PWM. AFM shows the number of times that each activity has appeared on each position in the alignment. The values in AFM (i.e. $A_{k,j}$) are calculated using equation 7.1. Value of $X_{i,j}$ in equation 7.1 is

Table 7.4: Absolute and relative relative frequency matrix.

	Activity Positions in Trace											
Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	
`a`	30.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`b`	0.0	10.0	10.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`c`	0.0	10.0	10.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`d`	0.0	10.0	10.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`e`	0.0	0.0	0.0	0.0	30.0	0.0	0.0	0.0	0.0	0.0	0.0	
`f`	0.0	0.0	0.0	0.0	0.0	30.0	0.0	0.0	0.0	0.0	0.0	
`g`	0.0	0.0	0.0	0.0	0.0	0.0	12.0	12.0	0.0	0.0	0.0	**(A) Absolute Frequency**
`h`	0.0	0.0	0.0	0.0	0.0	0.0	6.0	6.0	0.0	0.0	0.0	
`i`	0.0	0.0	0.0	0.0	0.0	0.0	6.0	6.0	0.0	0.0	0.0	
`j`	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	0.0	0.0	0.0	
`k`	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	24.0	0.0	0.0	
`l`	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	24.0	0.0	
`m`	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0	0.0	24.0	

`a`	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`b`	0	0.33	0.33	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`c`	0	0.33	0.33	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`d`	0	0.33	0.33	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
`e`	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
`f`	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
`g`	0.0	0.0	0.0	0.0	0.0	0.0	0.4	0.4	0.0	0.0	0.0	**(B) Relative Frequency**
`h`	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	
`i`	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.2	0.0	0.0	0.0	
`j`	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.0	0.0	
`k`	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	0.0	
`l`	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.8	0.0	
`m`	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.0	0.0	0.8	

calculated using 7.3.

$$A_{k,j} = \sum_{i=1}^{N} I(X_{i,j} = k) \tag{7.1}$$

Table 7.4(a) shows the AFM of online shopping process. For example, score of the trace <`a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`, `k`, `l`, `m`> using AFM evaluates to $30 \times 10 \times 10 \times 10 \times 30 \times 30 \times 12 \times 06 \times 24 \times 24 \times 24 = 2.6 \times 10^{13}$.

Table 7.5: Background normalized relative frequency matrix.

	Position specific scoring matrix											
Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	
a	0.86	0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.014	
b	0.014	0.29	0.29	0.29	0.014	0.014	0.014	0.014	0.014	0.014	0.014	
c	0.014	0.29	0.29	0.29	0.014	0.014	0.014	0.014	0.014	0.014	0.014	
d	0.014	0.29	0.29	0.29	0.014	0.014	0.014	0.014	0.014	0.014	0.014	
e	0.014	0.014	0.014	0.014	0.86	0.014	0.014	0.014	0.014	0.014	0.014	
f	0.014	0.014	0.014	0.014	0.014	0.86	0.014	0.014	0.014	0.014	0.014	**(A)**
g	0.011	0.011	0.011	0.011	0.011	0.011	0.35	0.35	0.011	0.011	0.011	**Relative frequency**
h	0.060	0.060	0.060	0.060	0.060	0.060	0.344	0.344	0.060	0.060	0.060	**(B=$\sqrt(N) = 5.47$)**
i	0.060	0.060	0.060	0.060	0.060	0.060	0.344	0.344	0.060	0.060	0.060	
j	0.003	0.003	0.003	0.003	0.003	0.003	0.172	0.003	0.003	0.003	0.003	
k	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.688	0.011	0.011	
l	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.688	0.011	
m	0.014	0.014	0.014	0.014	0.014	0.014	0.014	0.183	0.014	0.014	0.691	
a	0.996	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.003	
b	0.003	0.332	0.332	0.332	0.003	0.003	0.003	0.003	0.003	0.003	0.003	
c	0.003	0.332	0.332	0.332	0.003	0.003	0.003	0.003	0.003	0.003	0.003	
d	0.003	0.332	0.332	0.332	0.003	0.003	0.003	0.003	0.003	0.003	0.003	
e	0.003	0.003	0.003	0.003	0.996	0.003	0.003	0.003	0.003	0.003	0.003	
f	0.003	0.003	0.003	0.003	0.003	0.996	0.003	0.003	0.003	0.003	0.003	**(B)**
g	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.398	0.398	0.0002	0.0002	0.0002	**Relative Frequency**
h	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.20	0.20	0.0001	0.0001	0.0001	**(B=0.1)**
i	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.20	0.20	0.0001	0.0001	0.0001	
j	0.00006	0.00006	0.00006	0.00006	0.00006	0.00006	0.19	0.00006	0.00006	0.00006	0.00006	
k	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.79	0.0002	0.0002	
l	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.0002	0.79	0.0002	
m	0.003	0.003	0.003	0.003	0.003	0.003	0.003	0.199	0.003	0.003	0.79	

7.3.2 Relative Frequency Matrix (RFM)

Relative Frequency depicts the relative number of times a particular activity is observed with respect to other activities at a specific position in the trace alignment. An RFM is obtained by dividing the values in the AFM by the number of traces in the alignment, thereby normalizing the values.

Formally, given a set X of N aligned traces of length l, the elements of the RFM are calculated (using the equation 7.2).

$$R_{k,j} = \frac{1}{N}\sum_{i=1}^{N} I(X_{i,j} = k) \tag{7.2}$$

Where, $i \in (1, ..., N)$, $j \in (1, ..., l)$, k is the set of symbols in the alphabet and $I(a = k)$ is an indicator function where $I(a = k)$ is 1 if $a = k$ and 0 otherwise. Value

Table 7.6: Background normalized relative frequency matrix.

	Position specific scoring matrix											
Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	
a	8.95	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	
b	0.14	3.02	3.02	3.02	0.14	0.14	0.14	0.14	0.14	0.14	0.14	
c	0.14	3.02	3.02	3.02	0.14	0.14	0.14	0.14	0.14	0.14	0.14	
d	0.14	3.02	3.02	3.02	0.14	0.14	0.14	0.14	0.14	0.14	0.14	
e	0.14	0.14	0.14	0.14	8.95	0.14	0.14	0.14	0.14	0.14	0.14	
f	0.14	0.14	0.14	0.14	0.14	8.95	0.14	0.14	0.14	0.14	0.14	**(C)**
g	0.144	0.144	0.144	0.144	0.144	0.144	4.60	4.60	0.144	0.144	0.144	**Normalized Relative**
h	1.53	1.53	1.53	1.53	1.53	1.53	8.82	8.82	1.53	1.53	1.53	**Frequency**
i	1.53	1.53	1.53	1.53	1.53	1.53	8.82	8.82	1.53	1.53	1.53	**(B=$\sqrt{(n)}$)**
j	0.15	0.15	0.15	0.15	0.15	0.15	8.6	0.15	0.15	0.15	0.15	
k	0.144	0.144	0.144	0.144	0.144	0.144	0.144	0.144	9.05	0.144	0.144	
l	0.144	0.144	0.144	0.144	0.144	0.144	0.144	0.144	0.144	9.05	0.144	
m	0.14	0.14	0.14	0.14	0.14	0.14	0.14	1.90	0.14	0.14	6.35	
a	10.37	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	
b	0.031	3.45	3.45	3.45	0.031	0.031	0.031	0.031	0.031	0.031	0.031	
c	0.031	3.45	3.45	3.45	0.031	0.031	0.031	0.031	0.031	0.031	0.031	
d	0.031	3.45	3.45	3.45	0.031	0.031	0.031	0.031	0.031	0.031	0.031	
e	0.031	0.031	0.031	0.031	10.37	0.031	0.031	0.031	0.031	0.031	0.031	
f	0.031	0.031	0.031	0.031	0.031	10.37	0.031	0.031	0.031	0.031	0.031	**(D)**
g	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	5.2	5.2	0.0026	0.0026	0.0026	**Normalized Relative**
h	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	5.12	5.12	0.0025	0.0025	0.0025	**Frequency**
i	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	5.12	5.12	0.0025	0.0025	0.0025	**(B=0.1)**
j	0.003	0.003	0.003	0.003	0.003	0.003	9.5	0.003	0.003	0.003	0.003	
k	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	10.39	0.0026	0.0026	
l	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	0.0026	10.39	0.0026	
m	0.031	0.031	0.031	0.031	0.031	0.031	0.031	2.079	0.031	0.031	8.22	

of $X_{i,j}$ in equation 7.2 is calculated using 7.3.

$$X_{i,j} \begin{cases} 1 & \text{if } a = k \\ 0 & \text{otherwise} \end{cases} \tag{7.3}$$

Where, $i \in (1, ..., N)$, $j \in (1, ..., l)$, k is the set of symbols in the alphabet and $I(a = k)$ is an indicator function where $I(a = k)$ is 1 if $a = k$ and 0 otherwise.

Table 7.4(B) shows the relative frequency derived by applying equation 7.2. From the definition of RFM, it can be inferred that the sum of values for a specific position adds to 1. Each column in the RFM can, therefore, be considered as following an independent multinomial distribution. This makes it easy to calculate the probability of any trace given an RFM, by multiplying the relevant probabilities at each position. For example, score of the trace <`a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`, `k`, `l`, `m`> RFM evaluates to

Table 7.7: Position Weight Matrix.

	Position specific scoring matrix											
Activities	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	
a	3.16	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	
b	-2.83	1.67	1.67	1.67	-2.83	-2.83	-2.83	-2.83	-2.83	-2.8	-2.83	
c	-2.83	1.67	1.67	1.67	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	
d	-2.83	1.67	1.67	1.67	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	
e	-2.83	-2.83	-2.83	-2.83	3.16	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	
f	-2.83	-2.83	-2.83	-2.83	-2.83	3.16	-2.83	-2.83	-2.83	-2.83	-2.83	**(A)**
g	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	2.20	2.20	-2.79	-2.79	-2.79	**Log likelihood**
h	0.61	0.61	0.61	0.61	0.61	0.61	3.14	3.14	0.61	0.61	0.61	**(B=$\sqrt(n)$)**
i	0.61	0.61	0.61	0.61	0.61	0.61	3.14	3.14	0.61	0.61	0.61	
j	-2.71	-2.71	-2.71	-2.71	-2.71	-2.71	3.10	-2.71	-2.71	-2.71	-2.71	
k	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	3.17	-2.79	-2.79	
l	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	-2.79	3.17	-2.79	
m	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	-2.83	0.92	-2.83	-2.83	2.66	
a	3.37	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	
b	-5.01	3.45	3.45	3.45	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	
c	-5.01	3.45	3.45	3.45	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	
d	-5.01	3.45	3.45	3.45	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	
e	-5.01	-5.01	-5.01	-5.01	10.37	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	
f	-5.01	-5.01	-5.01	-5.01	-5.01	10.37	-5.01	-5.01	-5.01	-5.01	-5.01	**(B)**
g	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	2.37	2.37	-8.58	-8.58	-8.58	**Log likelyhoods**
h	-8.64	-8.64	-8.64	-8.64	-8.64	-8.64	2.35	2.35	-8.64	-8.64	-8.64	**Frequency**
i	-8.64	-8.64	-8.64	-8.64	-8.64	-8.64	2.35	2.35	-8.64	-8.64	-8.64	**(B=0.1)**
j	-8.38	-8.38	-8.38	-8.38	-8.38	-8.38	3.24	-8.38	-8.38	-8.38	-8.38	
k	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	3.37	-8.58	-8.58	
l	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	-8.58	3.37	-8.58	
m	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	-5.01	1.05	-5.01	-5.01	3.03	

$1 \times 0.33 \times 0.33 \times 0.33 \times 1 \times 1 \times 0.4 \times 0.2 \times 0.8 \times 0.8 \times 0.8 = 1.4 \times 10^{-3}$.

7.3.3 Effect of pseudocounts

A single deviation in a trace can knock the score down to zero. To address the problem of zeros, RFM is updated with pseudocounts (B) to reflect the overall composition of the traces to be considered. We set B to $\sqrt{N}$ (i.e. the square root of the total number of traces considered for constructing AFM and RFM) or 0.1 (regardless of the number of traces considered). In both scenarios, the effect of pseudocounts reduces

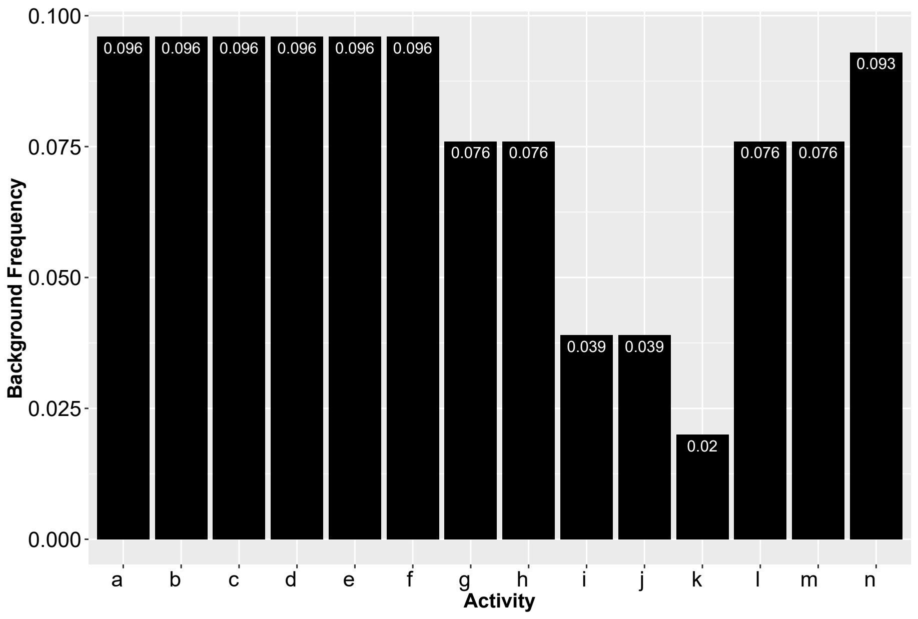


Figure 7.3: Activity Background Frequency.

as the increase in a number of traces ($\frac{\sqrt{N}}{N}$ in the first case, $\frac{0.1}{N}$ in the other).

The score for an individual activity at a particular position is equal to observed counts plus pseudocounts all divided by the total number of possible counts, as given in the equation 7.4.

$$U_{i,j} = \frac{z + x}{N + B} \tag{7.4}$$

Where,
z is the observed counts for the activity at the given position.
x is Pseudocount $\times$ overall frequency of activity.
N is total number of traces.
B is Pseudocount.

The value of B specific to each activity in the online shopping process is given in the fig. 7.3. It is calculated using the number of times activity appeared in the alignment to the total number of activities in the alignment. In the table 7.5(A), the value of the cell [a,1] is calculated as follows,

$$U_{i,j} = \frac{[30 + (\sqrt{2} \times 0.096)]}{30 \times \sqrt{2}}$$

Similarly, the values in the table 7.5(B) are calculated using B=0.1. The updated RFM with pseudocounts is shown in the table. 7.5(A-B). Normalized scores for pseudo count corrected matrix are obtained. Equation 7.5 is applied on pseudocounts shown in the table 7.5 (A - B). Normalised values are given in the table 7.6(C - D).

$$N = \frac{M_{k,j}}{b_k} \tag{7.5}$$

7.3.4 Creating Position Weight Matrix (PWM)

Most often the elements in PWMs are calculated as log likelihoods. Normalized values are transformed to log likelihood using the equation 7.6. Log likelihood values are shown in the table 7.7. Then, given a trace of length l and log-likelihood matrix, the score can be computed by adding the coefficients of the log-likelihood matrix corresponding to each activity in each position on the trace.

$$M_{k,j} = log_2(N) \tag{7.6}$$

7.4 Results and Discussions

In this section, we verify the proposed Position Weight Matrix on the random traces. The traces are constructed using the activities of the online process shown in fig. 7.1. We use the traces of length same as we used for constructing the alignment shown in the table 7.3.

Following is the test sequence used for verifying the result produced by Position Weight Matrix.

Test sequence:

`abcdefghklmadbcefigklmadbcefgiklmacbdefigklm`

The test sequence given above is scanned using windowing method with the size 11. We could see 34 different activity sequences out of the test sequences. The log-likelihood scores for each traces during each iteration is calculated using the values in the table 7.7.

Iteration 1:

`abcdefghklmadbcefigklmadbcefgiklmacbdefigklm`

Iteration 2:

`abcdefghklmadbcefigklmadbcefgiklmacbdefigklm`

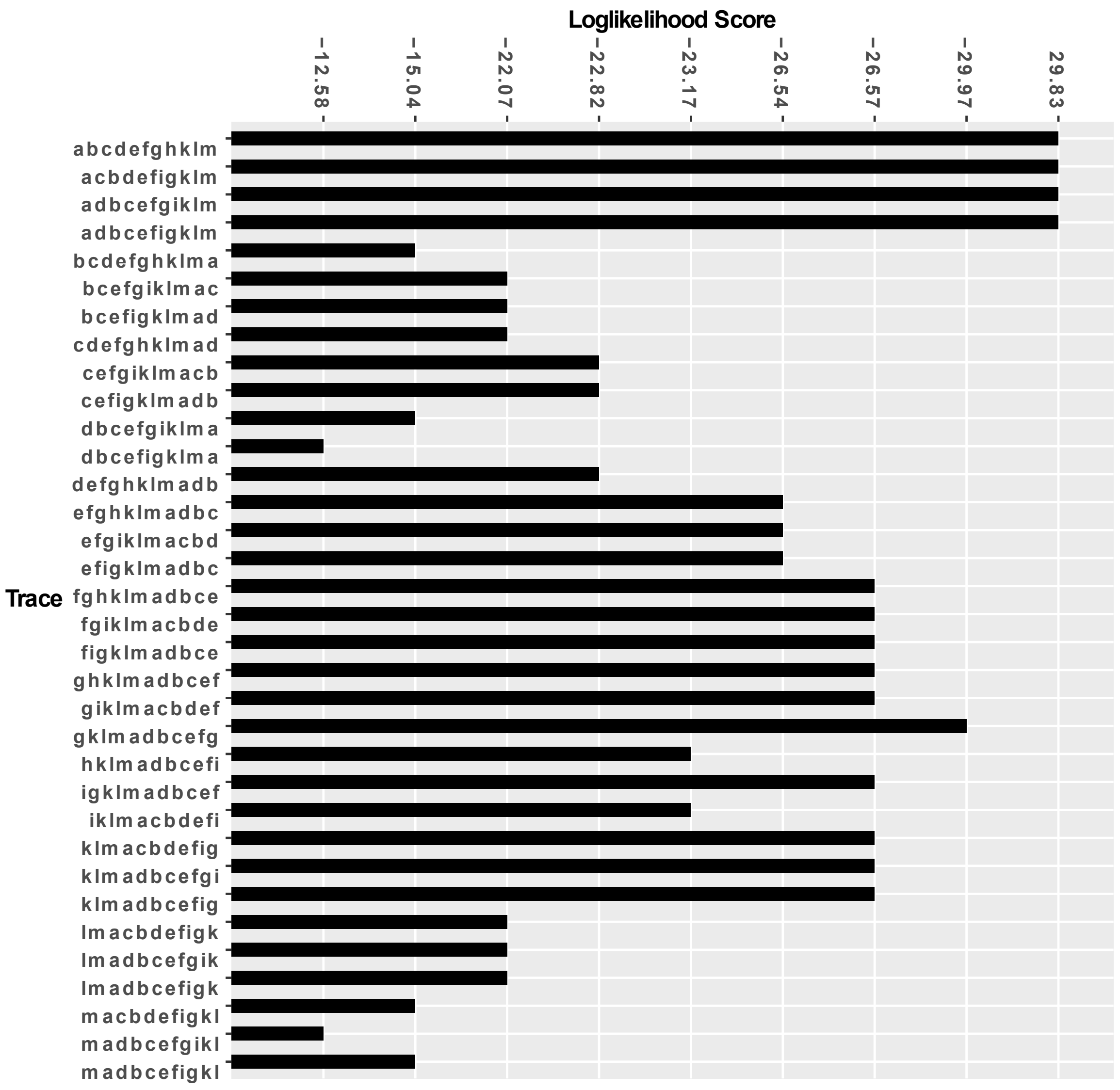


Figure 7.4: Log likelihood scores of test traces for B=$\sqrt{N}$.

Iteration 3:

abcdefghklmadbcefigklmadbcefgiklmacbdefigklm

...

...

...

Iteration 34:

abcdefghklmadbcefigklmadbcefgiklmacbdefigklm

The results of the experimental study has been plotted using bar graph and shown in the fig. 7.4 and 7.5. Fig. 7.4 shows the scores of traces using the PWM given in the table 7.7(A). Fig. 7.5 shows the scores of traces using the PWM given in the table 7.7(B).

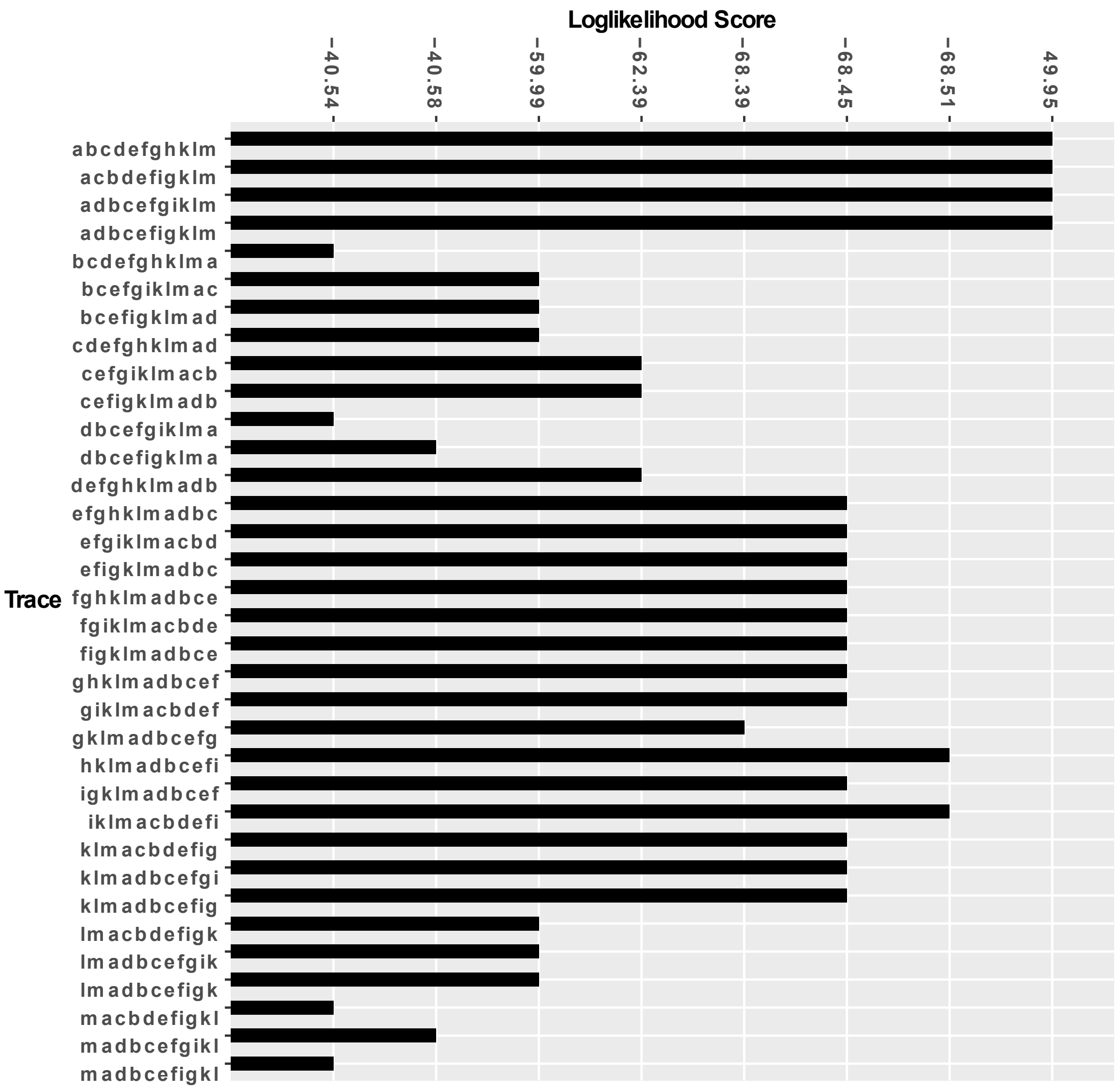


Figure 7.5: Log likelihood scores of test traces for B=0.1.

From the output shown in the fig 7.4 and 7.5 we can see that for the traces which formed the alignment have the considerably high log-likelihood score. Hence for any given sequence of control flow pattern with PWM at hand, we can calculate the likelihood score to decide whether it is a frequently used control flow execution procedure or not.

7.5 Summary

Identifying the frequent paths of execution in any operational process is a matter of interest for any organization, which could fetch a great benefit in terms of optimization of operations. This chapter presented a method for capturing the features related to control flow perspective, which assists in identifying and predicting the most frequently executed paths and helps in predicting the in control flow of a process and resource requirements.

The proposed method has been tested on real-life event log related to order process of the online shopping process, and it is proven to be efficient in terms of results obtained.

Chapter 8

Conclusions and Future Work

To conclude this thesis we first summarize some of the motivating factors that have driven the course of work presented in this thesis followed by our contributions. Although the techniques and concepts presented in this thesis take a step forward in addressing some of these factors, several challenges remain to be addressed to improve the applicability of process mining in general. We list some of these challenges and provide some directions for future work.

8.1 Summary of Contributions

The techniques proposed in this thesis for leveraging process mining practices can be used as a stand-alone tool. These methods can be thought as off-the-shelf entities for enhancing already available and upcoming process mining algorithms. Following are the brief descriptions of the contributions through this thesis,

- First contribution is concentrated on handling the phenomenon related to non-stationary learning problem called concept drift. We have presented a taxonomy for classifying different categories of concept drift based on problem characteristics. Further, a technique is presented for detecting and localizing the sudden concept drift in control flow perspective of the operational process.

- Proposed process logo notation for modeling control flow perceptive overcomes the drawbacks of traditional process mining algorithms. With the help of process logo, it is possible to illustrate various information in addition to the one possible with the traditional control flow visualization notations.

- We have presented a method for simplifying Spaghetti processes to discover a simplified Lasagna process. The method for identifying the possible and impossible paths of executions also has been proposed.
- We have presented a method for discovering and identifying the frequently executed paths of execution. With this information, organizations can optimize the paths which are carried out frequently and can predict the probable events in the due course of execution.

Following are some of the limitation to be solved in the work presented in this thesis.

- Our approach for handling concept drift deals only with detection and localization in control flow perspective. But, for addressing concept drift from end-to-end, techniques should consider all the aspects of phenomenon mentioned in fig. 3.2).
- Trace logo is an excellent notation for representing the control flow perspective of the process. It works fine with the traces of limited length. If it is applied on the longer and complex processes, it produces complex logos which are difficult to understand. One such example is given in fig. 5.10. It demands the techniques and methods to simplify the generated logos, to make it easily understandable.
- Finding the meaningful processes out of Lasagna control flow is an interesting problem. The method presented in this for the same, includes the generation. Still the proposed method can be optimized with the reduced feature set size.

8.2 Challenges and Directions for Future Work

In this section, we present some challenges. The reader is referred to the process mining manifesto (Van Der Aalst et al., 2012) for the list of additional problems in process mining.

- ***Online concept drift detection:*** Our methods concentrated on detecting concept drift in an offline setting, i.e., for postmortem analysis. Detecting concept drifts is essential for online analysis. We believe the proposed framework to be applicable even for online analysis. Few new challenges, however, emerge, e.g., the number of samples required remains an issue. Also, we need additional computational power and efficient techniques to do such analysis in near real time.

- ***Change process discovery:*** Organizations would be interested in discovering the evolution of change (e.g., as an animation depicting how the process has changed/evolved). Also, there are other applications such as deriving a configurable model for the process variants. A configurable process model describes a family of similar process models. The process variants discovered using concept drift can be merged to derive a configurable process model.

- ***Holistic approaches:*** We have discussed ideas on change detection and localization in the context of sudden changes to the control-flow perspective of a process. The data and resource perspectives are also, however, equally important. Features and techniques that can enable the detection of changes in other perspectives need to be discovered. Furthermore, there could be instances where more than one perspective (e.g., both control and resource) can change simultaneously. Hybrid approaches considering all aspects of change holistically need to be developed.

- ***Developing an application programming interface (APIs):*** It is at most necessary to develop APIs for assisting in the development of new process mining techniques. Process mining framework ProM plays a vital role in offering processes mining algorithms at one place, but it is a standalone tool, it does not allow the development of new method on the top of existing methods in it.

- ***Mining Lasagna trace clusters out of Spaghetti event log:*** We have presented a method in this thesis which is capable of extracting a single Lasagna process out of a given Spaghetti process. But, the method for clustering the traces of Spaghetti to mine a Lasagna are need.

- It is necessary to construct a repository of event logs, which should serve as a benchmark repository to test various process mining algorithms.

On a closing note, process mining had remarkable journey so far. The initial promise and success have led to more and more organizations look up to process mining as a solution for many different challenges. The contributions made in this thesis extends the journey by enabling process mining for handling concept drift and model simplification.

References

Adriansyah, A., Van Dongen, B. F. and Van Der Aalst, W. M. (2010). "Towards robust conformance checking." *Proc., International Conference on Business Process Management*. Springer, 122–133.

Agrawal, R., Gunopulos, D. and Leymann, F. (1998). "Mining process models from workflow logs." *Proc., International Conference on Extending Database Technology*. Springer, 467–483.

Alves de Medeiros, A., Van Dongen, B., Van Der Aalst, W. and Weijters, A. (2004). "Process mining: Extending the α-algorithm to mine short loops." Technical report, BETA Working Paper Series (WP 113), Eindhoven University of Technology, Eindhoven.

Antonio, R. (2016). "Business Report: The Data Made Me Do It, the next frontier for big data is the individual." https://www.technologyreview.com/s/514346/the-data-made-me-do-it/ (Jan. 20, 2016).

Arik Ragowsky, T. M. S. (2002). "Enterprise resource planning." *Journal of Management Information Systems*, 19(1), 11–15.

Bernard, M. (2017). "Big Data: 20 Mind-Boggling Facts Everyone Must Read." https://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/78762d4517b1 (Mar. 5, 2017).

Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA.

Bezerra, F., Wainer, J. and Van Der Aalst, W. M. (2009). "Anomaly detection using process mining." *Proc., International Conference on Enterprise, Business-Process and Information Systems Modeling*. Springer, 149–161.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F. and Wortman, J. (2008). "Learning bounds for domain adaptation." *Proc., International conference on advances in neural information processing systems*. 129–136.

Bose, R. J. C. and Van Der Aalst, W. M. (2009a). "Abstractions in process mining: A taxonomy of patterns." *Proc., International Conference on Business Process Management*. Springer, 159–175.

Bose, R. J. C. and Van Der Aalst, W. M. (2009b). "Context aware trace clustering: Towards improving process mining results." *Proc., 2009 SIAM International Conference on Data Mining*. SIAM, 401–412.

Bose, R. J. C. and Van Der Aalst, W. M. (2009c). "Trace clustering based on conserved patterns: Towards achieving better process models." *Proc., International Conference on Business Process Management*. Springer, 170–181.

Bose, R. J. C. and Van Der Aalst, W. M. (2010). "Trace alignment in process mining: opportunities for process diagnostics." *Proc., International Conference on Business Process Management*. Springer, 227–242.

Bose, R. J. C., Van Der Aalst, W. M. and Pechenizkiy, M. (2011). "Handling concept drift in process mining." *Proc., International Conference on Advanced Information Systems Engineering*. Springer, 391–405.

Bose, R. J. C., Van Der Aalst, W. M., Zliobaite, I. and Pechenizkiy, M. (2014). "Dealing with concept drifts in process mining." *IEEE Transactions on Neural Networks and Learning Systems*, 25(1), 154–171.

Carmona, J., Cortadella, J. and Kishinevsky, M. (2010). "New region-based algorithms for deriving bounded Petri nets." *IEEE Transactions on Computers*, 59(3), 371–384.

Carmona, J. and Gavalda, R. (2012). "Online techniques for dealing with concept drift in process mining." *Proc., International Conference on Advances in Intelligent Data Analysis*. Springer, 90–102.

Cook, J. E. and Wolf, A. L. (1998). "Discovering models of software processes from event-based data." *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(3), 215–249.

Datta, A. (1998). "Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches." *Information Systems Research*, 9(3), 275–301.

Day, W. H. and Edelsbrunner, H. (1984). "Efficient algorithms for agglomerative hierarchical clustering methods." *Journal of classification*, 1(1), 7–24.

De La Cruz, F., Paolini, M. A., Rothert, D. S. and Sethuraman, R. (2007). "Managing failover of J2EE compliant middleware in a high availability system." US Patent 7,246,256.

De Leoni, M. and Mannhardt, F. (2015). "Road Traffic Fine Management Process."

De Medeiros, A. K., Weijters, A. J. and Van Der Aalst, W. M. (2007). "Genetic process mining: an experimental evaluation." *Data Mining and Knowledge Discovery*, 14(2), 245–304.

Delany, S. J., Cunningham, P., Tsymbal, A. and Coyle, L. (2005). "A case-based technique for tracking concept drift in spam filtering." *Knowledge-Based Systems*, 18(4), 187–195.

Dennis, M. (2014). "Surprising Statistics About Big Data." http://www.baselinemag.com/analytics-big-data/slideshows/surprising-statistics-about-big-data.html (Sep. 13, 2014).

Dumas, M., Van Der Aalst, W. M. and Ter Hofstede, A. H. (2005). *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons.

Ferreira, D. R. and Gillblad, D. (2009). "Discovering process models from unlabelled event logs." *Proc., International Conference on Business Process Management*. Springer, 143–158.

Ferreira, H. M. and Ferreira, D. R. (2006). "An integrated life cycle for workflow management based on learning and planning." *International Journal of Cooperative Information Systems*, 15(04), 485–505.

Folino, F., Greco, G., Guzzo, A. and Pontieri, L. (2009). “Discovering expressive process models from noised log data.” *Proc., 2009 international database engineering & applications symposium.* ACM, 162–172.

Gaaloul, W., Baina, K. and Godart, C. (2005). “Towards mining structural workflow patterns.” *Proc., International Conference on Database and Expert Systems Applications.* Springer, 24–33.

Gama, J., Vzliobaite, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A. (2014). “A survey on concept drift adaptation.” *ACM Computing Surveys (CSUR)*, 46(4), 44.

Goedertier, S., Martens, D., Vanthienen, J. and Baesens, B. (2009). “Robust process discovery with artificial negative events.” *Journal of Machine Learning Research*, 10(6), 1305–1340.

Gold, E. M. (1967). “Language identification in the limit.” *Information and control*, 10(5), 447–474.

Greco, G., Guzzo, A. and Pontieri, L. (2008). “Mining taxonomies of process models.” *Data & Knowledge Engineering*, 67(1), 74–102.

Greco, G., Guzzo, A., Pontieri, L. and Sacca, D. (2006). “Discovering expressive process models by clustering log traces.” *IEEE Transactions on Knowledge and Data Engineering*, 18(8), 1010–1027.

Gunther, C. W., Rinderle-Ma, S., Reichert, M., Van Der Aalst, W. M. and Recker, J. (2008). “Using process mining to learn from process changes in evolutionary systems.” *International Journal of Business Process Integration and Management*, 3(1), 61–78.

Günther, C. W. and Van Der Aalst, W. M. (2007). “Fuzzy mining–adaptive process simplification based on multi-perspective metrics.” *Proc., International Conference on Business Process Management.* Springer, 328–343.

Günther and Anne Rozinat, C. W. (2012). “Disco: Discover Your Processes.” *Business Process Management.* 154–171.

Han, Y., Sheth, A. and Bussler, C. (1998). "A taxonomy of adaptive workflow management." *Proc., Workshop of the 1998 ACM Conference on Computer Supported Cooperative Work*. 1–11.

Hand, D. J. et al. (2006). "Classifier technology and the illusion of progress." *Statistical science*, 21(1), 1–14.

Harel, D. and Marelly, R. (2003). *Come, let us play: scenario-based programming using LSCs and the play-engine*, volume 1. Springer Science & Business Media.

Herbst, J. and Karagiannis, D. (2004). "Workflow mining with InWoLvE." *Computers in Industry*, 53(3), 245–264.

Jensen, K. and Kristensen, L. M. (2009). "Colored petri nets: Modelling and validation of concurrent systems." *Springer Verlag*, 978(3), 642.

Jiang, J. and Zhai, C. (2007). "Instance weighting for domain adaptation in NLP." *Proc., of the 45th International Conference Association of Computational Linguistics*, volume 7. 264–271.

Lamma, E., Mello, P., Montali, M., Riguzzi, F. and Storari, S. (2007). "Inducing declarative logic-based models from labeled traces." *Proc., International Conference on Business Process Management*. Springer, 344–359.

Leemans, S. J., Fahland, D. and Van Der Aalst, W. M. (2013a). "Discovering block-structured process models from event logs-a constructive approach." *Proc., International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer, 311–329.

Leemans, S. J., Fahland, D. and Van Der Aalst, W. M. (2013b). "Discovering block-structured process models from event logs containing infrequent behaviour." *Proc., International Conference on Business Process Management*. Springer, 66–78.

Liu, D. T. and Xu, X. W. (2001). "A review of web-based product data management systems." *Computers in industry*, 44(3), 251–262.

Luengo, D. and Sepúlveda, M. (2012). “Applying clustering in process mining to find different versions of a business process that changes over time.” *Proc., Business Process Management Workshops*. Springer, 153–158.

Luo, J., Pronobis, A., Caputo, B. and Jensfelt, P. (2007). “Incremental learning for place recognition in dynamic environments.” *Proc., 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 721–728.

Mannila, H. and Meek, C. (2000). “Global partial orders from sequential data.” *Proc., sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 161–168.

Măruşter, L., Weijters, A. T., Van Der Aalst, W. M. and Van Den Bosch, A. (2006). “A rule-based approach for process discovery: Dealing with noise and imbalance in process logs.” *Data mining and knowledge discovery*, 13(1), 67–87.

Michael, W. (2011). “ProM Package Documentation: KeyValue.” https://westergaard.eu/2011/07/prom-package-documentation-keyvalue/ (Jun. 22, 2017).

Needleman, S. B. and Wunsch, C. D. (1970). “A general method applicable to the search for similarities in the amino acid sequence of two proteins.” *Journal of molecular biology*, 48(3), 443–453.

Nerode, A. (1958). “Linear automaton transformations.” *Proceedings of the American Mathematical Society*, 9(4), 541–544.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Petri, C. A. (1962). “Kommunikation mit Automaten: Institut für Instrumentelle Mathematik.” *Schriften des IIM Nr*, 2.

Petter, B. B. (2015). “Big Data- for better or worse.” http://www.sintef.no/en/latest-news/big-data–for-better-or-worse/ (Dec. 23, 2016).

Regev, G., Soffer, P. and Schmidt, R. (2006). “Taxonomy of Flexibility in Business Processes.” *Proc., CEUR Workshop*, volume 236. CEUR-WS.org.

Reichert, M. and Weber, B. (2012). *Enabling flexibility in process-aware information systems: challenges, methods, technologies.* Springer Science & Business Media.

Rubin, V., Günther, C., Van Der Aalst, W. M., Kindler, E., Van Dongen, B. and Schäfer, W. (2007). "Process mining framework for software processes." *Software process dynamics and agility*, 169–181.

Rubin, V., Van Dongen, B., Kindler, E. and Günther, C. (2006). "Process mining: A two-step approach using transition systems and regions." *Proc., BPM Center Report.* Citeseer, 210–215.

Russell, N., Ter Hofstede, A. H., Edmond, D. and Van Der Aalst, W. M. (2004a). "Workflow data patterns." Technical report, QUT Technical report, Queensland University of Technology, Brisbane.

Russell, N., Ter Hofstede, A. H., Edmond, D. and Van Der Aalst, W. M. (2004b). "Workflow resource patterns." Technical report, BETA Working Paper Series, Eindhoven University of Technology, Eindhoven.

Russell, N., Ter Hofstede, A. H., Van Der Aalst, W. M. and Mulyar, N. (2006). "Workflow control-flow patterns: A revised view." *BPM Center Report*, 06–22.

Scheer, A.-W., Thomas, O. and Adam, O. (2005). "Process modeling using event-driven process chains." *Process-aware information systems*, 119–146.

Schimm, G. (2002). "Process miner - a tool for mining process schemes from event-based data." *Proc., European Workshop on Logics in Artificial Intelligence.* Springer, 525–528.

Schimm, G. (2004). "Mining exact models of concurrent workflows." *Computers in Industry*, 53(3), 265–281.

Schlimmer, J. C. and Granger, R. H. (1986). "Beyond Incremental Processing: Tracking Concept Drift." *Proc., International conference on Association for the Advancement of Artificial Intelligence.* 502–507.

Schlimmer, J. C. and Granger Jr, R. H. (1986). "Incremental learning from noisy data." *Machine learning*, 1(3), 317–354.

Schonenberg, H., Mans, R., Russell, N., Mulyar, N. and Van Der Aalst, W. M. (2008). "Process flexibility: A survey of contemporary approaches." *Advances in Enterprise Engineering I*. Springer, 16–30.

Shannon, C. E. (2001). "A mathematical theory of communication." *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1), 3–55.

Sheskin, D. J. (2007). *Handbook of Parametric and Nonparametric Statistical Procedures*. 4 edition. Chapman & Hall/CRC.

Song, M., Günther, C. W. and Van Der Aalst, W. M. (2009). "Trace clustering in process mining." *Proc., Business Process Management Workshops*. Springer, 109–120.

Song, M. and Van Der Aalst, W. M. (2007). "Supporting process mining by showing events at a glance." *Proc., of the 17th Annual Workshop on Information Technologies and Systems (WITS)*. 139–145.

Stanley, K. O. (2003). "Learning concept drift with a committee of decision trees." *Informe técnico: UT-AI-TR-03-302, Department of Computer Sciences, University of Texas at Austin, USA*.

Symons, M. J. (2014). "Customer relationship management (crm) systems." US Patent App. 14/192,692.

Ter Hofstede, A. H., Van Der Aalst, W. M., Adams, M. and Russell, N. (2009). *Modern Business Process Automation: YAWL and its support environment*. Springer Science & Business Media.

Thomas, S. (2015). "Permits in the Netherlands: an introduction for foreign Legal Counsel." https://www.akd.nl/en/b/Pages/Permits-in-the-Netherlands-an-introduction-for-foreign-Legal-Counsel.aspx (May 7, 2015).

Tsymbal, A. (2004). "The problem of concept drift: definitions and related work." Technical Report TCD-CS-2004-15, The University of Dublin, Trinity College, Department of Computer Science, Dublin, Ireland.

Tsymbal, A., Pechenizkiy, M., Cunningham, P. and Puuronen, S. (2008). "Dynamic integration of classifiers for handling concept drift." *Information fusion*, 9(1), 56–68.

Valmari, A. (1998). "The state explosion problem." *Lectures on Petri nets I: Basic models*, 429–528.

Van Der Aalst, W., Adriansyah, A., De Medeiros, A. K. A. and Arcieri, a. o. (2012). "Process mining manifesto." *Proc., Business process management workshops*. Springer, 169–194.

Van Der Aalst, W. M. (1998). "The application of Petri nets to workflow management." *Journal of circuits, systems, and computers*, 8(01), 21–66.

Van Der Aalst, W. M. (2001). "Re-engineering knock-out processes." *Decision Support Systems*, 30(4), 451–468.

Van Der Aalst, W. M. (2011). *Process mining: discovery, conformance and enhancement of business processes.* Springer Science & Business Media.

Van Der Aalst, W. M., Reijers, H. A., Weijters, A. J., Van Dongen, B. F., De Medeiros, A. A., Song, M. and Verbeek, H. (2007a). "Business process mining: An industrial application." *Information Systems*, 32(5), 713–732.

Van Der Aalst, W. M., Rosemann, M. and Dumas, M. (2007b). "Deadline-based escalation in process-aware information systems." *Decision Support Systems*, 43(2), 492–511.

Van Der Aalst, W. M., Rubin, V., Verbeek, H. M., Van Dongen, B. F., Kindler, E. and Günther, C. W. (2010). "Process mining: a two-step approach to balance between underfitting and overfitting." *Software and Systems Modeling*, 9(1), 87–111.

Van Der Aalst, W. M. and Song, M. (2004). "Mining social networks: Uncovering interaction patterns in business processes." *Proc., International Conference on Business Process Management*. Springer, 244–260.

Van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B. and Barros, A. P. (2003). "Workflow patterns." *Distributed and parallel databases*, 14(1), 5–51.

Van der Aalst, W. M., van Dongen, B. F., Herbst, J., Maruster, L., Schimm, G. and Weijters, A. J. (2003). “Workflow mining: A survey of issues and approaches.” *Data & knowledge engineering*, 47(2), 237–267.

Van Der Aalst, W. M. and Weijters, A. (2004). “Process mining: a research agenda.” *Computers in industry*, 53(3), 231–244.

Van Der Aalst, W. M., Weijters, T. and Maruster, L. (2004). “Workflow mining: Discovering process models from event logs.” *IEEE Transactions on Knowledge and Data Engineering*, 16(9), 1128–1142.

Van Der Aalst, W. M., Weske, M. and Grünbauer, D. (2005). “Case handling: a new paradigm for business process support.” *Data and Knowledge Engineering*, 53(2), 129–162.

Van derWerf, J. M. E., Van Dongen, B. F., Hurkens, C. A. and Serebrenik, A. (2009). “Process discovery using integer linear programming.” *Fundamenta Informaticae*, 94(3-4), 387–412.

Van Dongen, B. F. and Van Der Aalst, W. M. (2004). “EMiT: A process mining tool.” *Proc., International Conference on Application and Theory of Petri Nets*. Springer, 454–463.

Van Dongen, B. F. and Van Der Aalst, W. M. (2005). “Multi-phase process mining: Aggregating instance graphs into EPCs and Petri nets.” *Proc., PNCWB 2005 workshop*. 35–58.

Veck, M. (2016). “ProM: Process mining framework.” http://www.processmining.org/prom/start (Apr. 5, 2016).

Verbeek, H., Buijs, J. C., Van Dongen, B. F. and Van Der Aalst, W. M. (2010). “Xes, xesame, and prom 6.” *Proc., Conference on Advanced Information Systems Engineering (CAiSE)*. Springer, 60–75.

Waterman, M. S. (1995). *Introduction to computational biology: maps, sequences and genomes*. CRC Press.

Weber, B., Rinderle, S. and Reichert, M. (2007). “Change patterns and change support features in process-aware information systems.” *Proc., International Conference on Advanced Information Systems Engineering.* Springer, 574–588.

Weijters, A., Van Der Aalst, W. M. and De Medeiros, A. A. (2006). “Process mining with the heuristics miner-algorithm.” *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166, 1–34.

Weijters, A. J. and Van Der Aalst, W. M. (2003). “Rediscovering workflow models from event-based data using little thumb.” *Integrated Computer-Aided Engineering*, 10(2), 151–162.

Welytok, J. G. (2006). *Sarbanes-Oxley for dummies.* John Wiley & Sons.

Wen, L., Van Der Aalst, W. M., Wang, J. and Sun, J. (2007). “Mining process models with non-free-choice constructs.” *Data Mining and Knowledge Discovery*, 15(2), 145–180.

Wen, L., Wang, J., Van Der Aalst, W. M., Huang, B. and Sun, J. (2009). “A novel approach for process mining based on event types.” *Journal of Intelligent Information Systems*, 32(2), 163–190.

White, S. A. et al. (2004). “Business process modeling notation.” *Specification, BPMI. org*, 21.

Widmer, G. and Kubat, M. (1993). “Effective learning in dynamic environments by explicit context tracking.” *Proc., International Conference on Machine learning.* Springer, 227–243.

Widmer, G. and Kubat, M. (1996). “Learning in the presence of concept drift and hidden contexts.” *Machine learning*, 23(1), 69–101.

Wu, J., Ding, D., Hua, X.-S. and Zhang, B. (2005). “Tracking concept drifting with an online-optimized incremental learning framework.” *Proc., 7th ACM SIGMM international workshop on Multimedia information retrieval.* ACM, 33–40.

Yang, S. and Yao, X. (2008). “Population-based incremental learning with associative memory for dynamic environments.” *IEEE Transactions on Evolutionary Computation*, 12(5), 542–561.

Yen, V. C. (2003). “Technologies & Methodologies for Evaluating Information Technology in Business.” chapter Systems Analysis with Workflow Modeling. IGI Global, Hershey, PA, USA, 143–159.

Zliobaite, I. (2010). “Learning under Concept Drift: an Overview.” *CoRR*, abs/1010.4784.

List of Publications

Journal Publications

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2016). "Concept Drifts Detection and Localisation in Process Mining", *Information Journal*, 19(10B), 4611- 4616.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2016). "Best Resource Recommendation for a Stochastic Process", *Information Journal*, 19 (10B), 4617- 4622.

- **Manoj Kumar, M. V.**, Likewin Thomas Annappa, B., and Bartosz Krawczyk. "Aligning Traces to Detect and Localize Concept Drifts in Process Mining", *IEEE Transactions on Knowledge and Data Engineering* **(Communicated)**.

Conference Publications

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2014). "Phenomenon of concept drift from process mining insight." *Proc., 2014 IEEE Advance Computing Conference (IACC)*, IEEE , Gurgaon, India, 517-522.

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2015). "Capturing the Sudden Concept Drift in Process Mining". *Proc., 36th International Conference on Application and Theory of Petri Nets and Concurrency15th International Conference on Application of Concurrency to System Design (Petri Nets 2015 // ACSD 2015)*, CEUR Proceedings, University libre de Bruxelles, Brussels, Belgium, 131-143.

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2016). "Trace Logo: A Notation for Representing Control-Flow of Operational Process". *Proc., 18th International Conference on Management, Economics and Business Information (ICMEBI).* International Journal of Business and Economics Engineering, Tokyo, Japan, 3(5).

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2017). "Predicting frequent Execution path in information systems", *Proc., 2017 Second Inter-*

national Conference on Electrical, Computer and Communication Technologies (ICECCT). Coimbatore, IEEE.

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2017). "Distilling Lasagna from Spaghetti Processes." *Proc., 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI).* Hong Kong, ACM, 157-161.

- **Manoj Kumar, M. V.**, Likewin Thomas and Annappa, B. (2017). "Simplifying Spaghetti Processes to Find the Frequent Execution Paths". *Proc., International Conference on Smart Systems, Innovation and Computing (SSIC).* Springer LNCS, Manipal University, Jaipur, April 2017.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2014). "Efficient process mining through critical path network analysis." *Proc., Advance Computing Conference (IACC)*, IEEE , Gurgaon, India, 511-516.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2015). "An Optimal Process Model for a Real Time Process." *Proc., 36th International Conference on Application and Theory of Petri Nets and Concurrency15th International Conference on Application of Concurrency to System Design (Petri Nets 2015 // ACSD 2015),* CEUR Proceedings, University libre de Bruxelles, Brussels, Belgium, 117-131.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2016). "Optimized Path Recommendation for a Real Time Process". *Proc., 18th International Conference on Management, Economics and Business Information.* International Journal of Business and Economics Engineering, Tokyo, Japan, 3(5).

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2016). "Discovery of optimal neurons and hidden layers in feed-forward Neural Network". *Proc., Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech).* IEEE, Mauritius, 286-291.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2017). "An online decision support system for recommending an alternative path of execution."

Proc., 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). Coimbatore, IEEE.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2017). "Recommending an alternative path of execution using an online decision support system." *Proc., 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (ISMSI'17).* Hong Kong, ACM, 108-112.

- Likewin Thomas, **Manoj Kumar, M. V.** and Annappa, B. (2017). "Prediction of Gallstone Disease Progression using Modified Cascade Neural Network." *Proc., International Conference on Smart Systems, Innovation and Computing (SSIC).* Springer LNCS, Manipal University, Jaipur.

Appendices

Appendix A

Event logs used in experiments

Following are the links for downloading the experimental event log and the programs used for obtaining the results displayed in this thesis. The work in the thesis can be reproduced by others using the information provided in the following links.

- Chapter 4: Detecting and localizing control flow concept drift in process mining.
  `http://www.rectopage.com/manoj/concept_drift/`

- Chapter 5: Process Logo: Informative Visualization of Control Flow Perspective.
  `http://www.rectopage.com/manoj/process_logo`

- Chapter 6: Distilling Lasagna from Spaghetti Processes.
  `http://www.rectopage.com/manoj/lasagana_from_spaghetti`

- Chapter 7: Identifying Frequent Execution Paths in Information System.
  `http://www.rectopage.com/manoj/frequent_paths`

Brief Bio-Data

Manoj Kumar M V

Research Scholar

Department of Computer Science and Engineering

National Institute of Technology Karnataka, Surathkal

P.O. Srinivasnagar

Mangalore - 575025

Phone: +91 9164460178

Email: manojmv24@gmail.com

Permanent Address

Manoj Kumar M V

S/o M L Vasantha Kumara

Pethe Beedhi

Malebennur - 577530

Harihara (Tq.), Davanagere (Dist.)

Karnataka, INDIA

Qualification

M. Tech. in Computer Science and Engineering, Visvesvaraya Technological University, Belgaum, Karnataka, 2012.

B. E. Computer Science and Engineering, Visvesvaraya Technological University, Belgaum, Karnataka, 2010.