

A machine learning approach to understand business processes

Citation for published version (APA):

Maruster, L. (2003). *A machine learning approach to understand business processes*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Industrial Engineering and Innovation Sciences]. Technische Universiteit Eindhoven. <https://doi.org/10.6100/IR568241>

DOI:

[10.6100/IR568241](https://doi.org/10.6100/IR568241)

Document status and date:

Published: 01/01/2003

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

A machine learning approach to understand business processes

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op woensdag 27 augustus 2003 om 16.00 uur

door

Laura Mărușter

geboren te Baia Mare (Roemenië)

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. J.C. Wortmann
en
prof.dr. W.M.P. Daelemans

Copromotor:
dr. A.J.M.M. Weijters

A machine learning approach to understand business processes

Laura Mărușter

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Mărușter, Laura

A machine learning approach to understand business processes/ by Laura Mărușter.

– Eindhoven : Technische Universiteit Eindhoven, 2003. – Proefschrift.

ISBN 90-386-1688-0

NUR 982

Keywords: Process modelling / Machine learning / Knowledge discovery / Clustering

/ Process discovery (mining) / Petri nets

Printed by Universiteitsdrukkerij Technische Universiteit Eindhoven

Cover Illustration: M.C. Escher's "Relativity" ©2003 Cordon Art - Baarn - Holland.

All rights reserved.

Dedication

*To the Fish, which is mute, expressionless,
and does not think, but knows everything.*

Contents

I	Prolegomena	1
1	Introduction	3
1.1	Motivation for this research	3
1.2	Research statement	4
1.3	Scope of this thesis	4
1.4	Methodology	5
1.5	Thesis structure	6
2	Modelling a process from data	9
2.1	Process modelling	9
2.2	Inducing models by learning from data	10
2.2.1	Learning from data	10
2.2.2	Machine learning methods	12
2.3	Research design	17
II	Process modelling via clustering aggregated measures	21
3	Modelling the process of multi-disciplinary patients	23
3.1	Introduction	24
3.2	Understanding the problem domain	25
3.2.1	The medical problem	25
3.2.2	The logistic problem domain	26
3.3	Data collection and preparation	26
3.3.1	Data selection	27
3.3.2	Missing data	27
3.4	The aggregation of raw data	28
3.5	Development of logistic patient groups	29
3.5.1	Clustering experiments	30
3.5.2	Clustering experiment involving all logistic variables	30
3.5.3	Clustering experiment involving two latent factors	32
3.6	Development of predictive models	35
3.6.1	Experiment “all diagnoses”	36
3.6.2	Experiment “chronic diagnoses”	37
3.7	Discussion	42

3.8	Conclusions	43
III	Discovering a process from sequence data	45
4	The formal approach	47
4.1	Introduction	47
4.2	Classical Petri nets. Workflow nets.	48
4.2.1	Petri nets	48
4.2.2	Workflow nets	51
4.3	The discovery problem	52
4.3.1	The algorithm for discovering process models	55
4.3.2	Which processes can be rediscovered?	57
4.4	Process discovery literature	61
4.5	Conclusions	64
5	The practical approach	65
5.1	Introduction	65
5.2	Problem statement	66
5.3	Experimental setting and data generation	67
5.4	Discovering the log-based relations	69
5.4.1	The dependency/frequency table	69
5.4.2	The log-based relations	69
5.4.3	The relational metrics	71
5.5	Inducing models for discovering log-based relations	73
5.5.1	The logistic regression model	74
5.5.2	The rule-based model	77
5.5.3	Evaluating the induced models	80
5.6	The influence of log characteristics on the rule-based model performance	83
5.7	Conclusions	87
6	Applications	89
6.1	Discovering business processes from simulated logs	90
6.1.1	Data considerations	90
6.1.2	An application to a product development process	94
6.1.3	Discussion	101
6.2	Discovering the process of handling fines	102
6.2.1	The COLLECTION sub-process	102
6.2.2	The UNDELIVERABLE LETTER RETURN (ULR) sub-process	104
6.2.3	Discussion	106
6.3	Discovering the treatment process of multi-disciplinary patients	106
6.3.1	Applying the discovery method to multi-disciplinary patient data	106
6.3.2	Discussion	108
6.4	Conclusions	110

IV	Conclusions	113
7	Conclusions and suggestions for further research	115
7.1	Contributions of this thesis	115
7.1.1	The use of aggregated data	115
7.1.2	The use of sequence data	116
7.1.3	Combining the two approaches	118
7.2	Further research	119
A	Discovering process models from simulated ADONIS logs	123
B	The rule sets that characterize the logistic clusters	131
C	The rule set for detecting causal relations	137
	References	139
	Acknowledgements	147
	Summaries	149
	Summary	149
	Samenvatting	151
	Rezumat	153
	Curriculum vitae	155

Part I

Prolegomena

Chapter 1

Introduction

1.1 Motivation for this research

In current organizations and businesses, a continuous push can be witnessed for increased diversification of services and products. This requires increasingly efficient and effective organizations and production environments. Business processes need to be appropriately organized, preferably according to scientific methods. Since the principles of Scientific Management were introduced (Taylor [1947]), Business Administration has become an established scientific discipline focusing on the subject of organizing work, usually within an organizational context. With the increasing involvement of information processing in organizations, the field of Business Process Management emerged.

Business Process Management can be seen as the field of designing and controlling business processes (Reijers [2002]). The *design* dimension focuses on strategic decisions like, for example, the restructuring of a business process. The *control* of a business process focuses more on decisions that are taken on the real-time, operational and tactical level of decision making, like for example production planning, resource allocation and budgeting. The challenge for the managers and persons responsible for the process is to design and control processes efficiently and effectively. To achieve this goal, it is of critical importance to have a good understanding of the business process.

Because substantial amount of business process information is recorded electronically, the acquired data can be helpful to gain a clear picture of the business process. The main idea of this thesis is to employ machine learning techniques to provide methods for an understanding of processes using data.

When designing a control system, the process should be known, and certain design principles should be followed. For example, to design a control system for a hospital, production control approaches from manufacturing are used. Because a hospital is not a manufacturing organization, specific concepts such as, what is the “product”, need to be elaborated. In such a situation, patient histories provide information that is used to characterize the hospital process. As soon as the hospital process is characterized, a control system may be developed to coordinate the patient volumes.

In order to design a system that supports the business process, a designer has to

construct a detailed model accurately describing the process. Modelling a process is a complex task because it requires extensive knowledge of the process at hand (i.e. long discussions with the workers and the management are needed), it is time consuming and often prone to subjectivity. One of the main ideas of this thesis is to collect data at runtime to support process design and analysis. The information collected at run-time can be used to derive a model automatically, explaining the events recorded. Our attempt is to employ techniques from the realm of machine learning to obtain insights into business processes. We call this approach *process discovery*.

The topic of process discovery is related to management trends such as Business Process Reengineering (BPR), Business Intelligence (BI), Business Process Analysis (BPA), Continuous Process Improvement (CPI) and Knowledge Management (KM). Process discovery can be used as an input model for BPR and CPI activities. However, process discovery should not be seen as a tool to (re)design processes. The goal is to understand what is actually going on in reality. Despite the fact that process discovery is not a tool for designing processes, it is evident that a good understanding of the existing processes is vital for any redesign effort.

1.2 Research statement

The research objective of this thesis is to provide methods that allow us to understand business processes by applying machine learning techniques. The basic material that we use are empirical data, i.e. the recorded histories of process events that occurred over time. The goal of having deeper knowledge about processes is two-sided: to provide scientific knowledge, that is to “to develop laws and theories that explain, predict, understand, and control phenomena” (Hunt [1991]), and to enable practical applications, i.e. to have a solid base to develop systems that efficiently support and control business processes.

From the research objective, we derived five main research questions:

1. What data representations can be useful for modelling business processes?
2. How can machine learning techniques be used for the clustering of process-related measures?
3. Knowing that relevant clusters can be developed, how can they be used to make predictions?
4. What kind of processes can be discovered from past process executions?
5. Is it possible to extract process models from data?

1.3 Scope of this thesis

In this thesis we show that machine learning techniques can be used successfully to understand a process on the basis of data, by means of clustering process-related measures, induction of predictive models, and process discovery.

We can target the analysis of two sorts of data, namely aggregated data and sequence data.

Aggregated data result from some transformations of raw data, focusing on a specific concept, that is not yet explicit in the raw data. This aggregation is similar to *feature construction*, as used in the machine learning domain. Feature construction is to transform the original representation in a new, usually more compact representation, that captures most (or the most relevant) of the original characteristics. In this thesis, aggregated data are the variables that result from operationalizing the concept of process complexity. These aggregated data are used to develop *logistic* homogeneous clusters. This means that elements in different clusters differ from the routing complexity point of view. We show that developing homogeneous clusters for a given process is relevant in connection with the induction of predictive models. The routing in the process can be predicted using the logistic clusters. We do not aim to provide concrete directives for building control systems, rather our models should be taken as indications of their potential.

Sequence data describe the sequence of activities over time in a process execution. They are recorded in a process log, during the execution of the process steps. Due to exceptions, missing or incomplete registration and errors, the data can be noisy. By using sequence data, the goal is to derive a model explaining the events recorded. In situations without noise and when sufficient information is available, we provide a method for building a process model from the process log. Moreover, we discuss the class of models for which it is possible to accurately rediscover the model by looking at the process log. Machine learning techniques are especially useful when discovering a process model from noisy sequence data. Such a model can be further analyzed and eventually improved, but these issues are beyond the scope of this thesis.

Through the applications of our proposed methods on different data, we have shown that our methods result in useful models and subsequently can be used in practice. We applied our methods on data sets for which (i) it was possible to aggregate relevant information and (ii) sequence data were available. Only in one application it was possible to test both approaches, while in another two applications we only have tested the use of sequence data.

1.4 Methodology

In this section we indicate some important methodological considerations. In Chapter 2 we provide more details about our modelling methodology.

The research approach used in our thesis is the *inductive method*. The inductive method bases itself on observations in the real world, aiming to find laws and theories about these observations. Contrary to the inductive approach, the *deductive method* starts from axioms with the goal of arriving at theorems that logically follow from them. In the deductive method, logic is the authority. If a statement logically follows from the axioms of the system, it must be true. In the inductive method, observations from the real world are the authority. If a reasoning conflicts with what happens in the real world, the reasoning must be changed or abandoned. The process of applying the inductive method is called *inductive inference* and can be defined as the “process of hypothesizing a general rule from examples” (Angluin and Smith [1983]).

In this thesis the inductive inference method is employed, i.e. we use machine learning techniques to induce models from data. To strengthen our results, we also

employ the deductive method to develop models that can be formally analyzed.

The main concerns when performing quantitative modelling are the *validity*, *generalizability* of the findings and *reliability* of the collection and analysis of data.

We plan to test the *validity* of our results by checking the *internal* and *external validity* (Bryman [1988]). We use the term of *internal validity* in the sense that internal validity is certifying the degree to which the causes and effects identified are actually supported by the data, rather than secondary consequences of some other relationships. For example, internal validity is certified when a researcher controls all extraneous variables and the only variable influencing the results of a study is the one being manipulated by the researcher. This means that the variable the researcher intended to study is indeed the one affecting the results and not some other, unwanted variables. *External validity* concerns with the general applicability of the conclusions, i.e. shows to what extent these conclusions reflect reality and can be demonstrated in other situations. This refers to the extent to which the results of a study can be generalized or extended to others.

To test the ability of our method to scale up under conditions of increasing difficulty, we occasionally develop models from simulated data. The advantage of performing experiments using simulation is that it is easier to control extraneous variables, insuring a better internal validity. The models are externally validated by testing the model against data resulting from different domains: the experts responsible for the investigated process are asked about the validity of the results. By validating the models against data from different domains there is support for generalizable and replicable findings.

To assess the quality of the induced models with machine learning techniques, specific performance measures such as classification accuracy, and data sampling methods such as n -fold cross-validation, etc. are used. Further considerations about the inducing of valid machine learning models are discussed in Chapter 2.

Reliability refers to the repeatability of the results, independent of the investigator in the cases. This refers to the replication of the results by repeating the same case, with another investigator. To ensure reliability, we elaborate a well-documented case procedure and report all the steps involved.

1.5 Thesis structure

This thesis has four parts.

In Part I (Chapter 1 and Chapter 2) the context, the boundaries and the research objectives of the thesis are introduced. In Chapter 2, some considerations are discussed about process modelling, inducing models by learning from data and the methodology of inducing models using machine learning techniques. Part I concludes with presenting the framework of the research design. Models can be learned from data if data are represented as aggregated data and sequence data, which subsequently result in two research perspectives.

In Part II the first research perspective is developed, i.e. process modelling by clustering aggregated measures. Employing aggregated data, processes can be modelled using machine-learning methods. In Chapter 3, the modelling of the logistic patient-care process by constructing homogeneous patient groups is presented. Based

on a-priori patient information (e.g. age, gender and chronic diagnosis) and the patient groups, a rule-based model can be induced to predict the future route of patients inside the hospital.

Part III refers to the second research perspective of this thesis, i.e. the discovery of a process model from sequence data. Two distinct approaches are investigated. In Chapter 4 we present the theoretical approach, where we show that if there is sufficient information and there is no noise, a process model can be built from the process log. Moreover, the class of models for which it is possible to accurately rediscover the model on the basis of the process log is described. In Chapter 5 a practical approach is presented: machine learning techniques are employed to construct the process model from noisy and incomplete sequence data. Chapter 6 further validates our methods by means of three applications. In the first one, we describe the application of our discovery technique on simulated data taken from a business process management modelling tool, where the process model is known. In the second application we discuss the discovery of the process model from real data of a Dutch governmental organization. Finally, in the third application, we use the hospital data from Chapter 3 to discover the underlying process of treating multi-disciplinary patients.

Part IV concludes the thesis with a discussion about its contributions. Finally, some suggestions for further research are given.

Chapter 2

Modelling a process from data

In the previous chapter the motivation of modelling a process by using data was briefly introduced. In this chapter we present more details about process modelling and inducing models by learning from data. The methodology of inducing models using machine learning techniques is discussed. The research design used in this research ends this chapter.

2.1 Process modelling

The notion of *model* is formally developed in a discipline of mathematics, called *Model Theory* (Tarski [1947]), which shows how to apply logic to the study of structures in pure mathematics. Rather than developing an abstract concept, we share the view of empirical sciences, where a *model* is defined as “a more or less mathematized array of specific concepts and principles intended to represent some essential aspects of a particular piece of empirical reality” (Moulines [2002]). We base our view about model characteristics on the following model’s characteristics specified in Moulines [2002]:

(i) a model is not supposed to cover all aspects of the empirical domain depicted by the model, (ii) a model it is not supposed to provide the ultimate truth about the domain in question and (iii) the acceptance of the model allows for successive revisions, refinements, suppressions, additions, etc.

In this thesis, we understand the notion of *process* as “the set of partially ordered process steps intended to reach a goal” (Humphrey and Feiler [1992]). The *process steps* are elementary (atomic) processes such as task, event, state, executable activity, functional operation. A *process description* is a written description of the intended or predicted behavior of the process. A *process model* is a description using a formal language of an actual or proposed process that represents the process elements. A *process enactment* is an instance of one executed process. Process models are developed to gain understanding, to analyze and finally to enact them in concrete situations. In the context of this thesis, we want to refer to generic processes, that are not domain dependent.

Different process modelling perspectives exist, depending on the domain. For example, in enterprise modelling, models are developed to represent the structure and behavior of a business entity accurately. The purpose of such models is “to provide common understanding among users about enterprise operations and structure, to support analysis or decision-making, or to control operations of the enterprise” (Veradat [1996]). From the perspective of software processes modelling, a process “is a set of partially ordered steps intended to reach a goal” (Humphrey and Feiler [1992]). In Artificial Intelligence, the issues of problem solving and planning are modelled as a process of searching for routes, i.e. sequences of actions, that lead to a solution or a goal state (Luger and Stubblefield [1993]).

2.2 Inducing models by learning from data

2.2.1 Learning from data

The concept of learning is hard to define. In general it is assimilated with terms as “to gain knowledge, or understanding, by study, instruction, or experience”. Trying to understand how humans learn, researchers from artificial intelligence attempt to develop methods for accomplishing the acquisition and application of knowledge algorithmically (i.e. on computers), naming this *machine learning*. According to Mitchell’s definition, “machine learning is concerned with the question how to construct computer program that automatically improve with experience” (Mitchell [1995]). Developing computer programs that “learn” requires knowledge from many fields. Therefore, in machine learning, concepts and results from many fields can be found, including statistics, artificial intelligence, philosophy, information theory, biology, cognitive science, computational complexity, and control theory.

A natural question arises: why would machines learn, when they can be designed from the beginning to perform as desired? Apart from the reason that it may provide explanations about how humans (and animals) learn, there are important engineering reasons for machines to learn. In his machine learning notes, Nilsson mentions the following lessons (Nilsson [2001]):

- Some tasks cannot be defined well except by example; that is, we might be able to specify input/output pairs, but not a concise relationships between inputs and the desired outputs. We would like machines to be able to adjust their internal structure to produce correct outputs for a large number of sample inputs and thus suitably constrain their input/output function to approximate the relationship implicit in the examples.
- It is possible that important relationships and correlations are hidden in large piles of data. Machine learning methods can often be used to extract these relationships.
- Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might be not completely known at design time. Machine learning methods can be used for on-the-job improvement of existing machine designs.

- The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down.
- Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign.
- New knowledge about tasks is constantly being discovered by humans. Vocabulary changes. There is a constantly stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning might be able to track much of it.

Different types of learning can be identified. Dietterich divides learning in four areas (Dietterich [1990]): rote learning (that is memorization by repetition), learning by being told, learning from examples and learning by analogy. Learning by examples is employing *inductive inference*, in the sense that general concepts are resulting from particular examples (Angluin and Smith [1983]).

Inductive inference is used as common mechanism in knowledge discovery, data mining, machine learning, statistics, etc., namely in disciplines that concern learning from data. These disciplines are not mutually exclusive, as emerges from the following well-known definitions, presented below.

Fayyad et al. define *Knowledge discovery in databases* (KDD) as “the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data” (Fayyad et al. [1996]). Knowledge discovery can be viewed as a multidisciplinary activity that exploits artificial intelligence (machine learning, pattern recognition, expert systems, knowledge acquisition) and mathematics disciplines (statistics, theory of information, uncertainty processing).

Data mining is “a step in the KDD process consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over data” (Fayyad et al. [1996]). Data mining is often pattern-focused rather than model-focused, i.e. rather than building a coherent global model which includes all variables of interest, data mining algorithms would typically produce a set of statements about local dependencies among variables.

In Data Mining and KDD, machine learning is most commonly used to mean the application of induction algorithms. Machine Learning is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to “learn”, whereas data mining is more an engineering field (Bruha [2001]).

Data mining and KDD have a lot in common with statistics. In statistics the methods are developed to describe relations between variables for prediction, quantifying effects or suggesting causal paths. Such methods use probability calculus to quantify the uncertainty associated with drawing inferences from data.

Fayyad et. al define two goals of the KDD process: (i) the **Verification** of the user’s hypothesis and (ii) the **Discovery** of the new patterns (Fayyad et al. [1996]). The **Discovery** goal can be further divided into **Prediction**, where new patterns are found for the purpose of predicting the future behavior of some entities and **Description**, where patterns can be found for the purpose of presenting them to a user in an understandable form (Fayyad et al. [1996]).

The view used in this thesis is to employ machine learning and statistical techniques for knowledge discovery in business data. We are interested to induce process models on the basis of data to provide useful insights into processes. Namely, we want to obtain models for (i) *predicting* the future behavior of the process's cases and (ii) providing a *description* of the process in an adequate formalism.

2.2.2 Machine learning methods

In the paradigm of learning from examples, one possible representation is a vector of *attributes* (features, variables) describing *examples* (instances, objects) (Mitchell [1995]). One of the basic machine learning task is *classification*: to map examples into predefined groups or *classes*. This task is often referred as *supervised* learning, because the classes are determined before examining the data. Given a training set of data and correct classes, the computational model successively applies each entry in the training set. Based on its ability to handle each of these entries correctly, the model is changed to ensure that it works better with this entry if it were applied again. Given enough input values, the model will learn the correct behavior for any potential entry. Machine learning algorithms as decision trees, neural networks and genetic algorithms are examples of supervised learning algorithms.

In the *unsupervised* learning approach, models are built from data without predefined classes. The goal is to organize, or to reduce dimensionality of the unlabelled data to obtain insights about internal data structure. Data instances are grouped together using a certain similarity metric. With the help of some evaluation methods, a decision can be made about the meaning of the formed clusters. Examples of unsupervised learning algorithms are k-means clustering and self organizing maps.

In this thesis we show that combinations of machine learning and statistical techniques are useful for providing insights into processes. For this, we use decision trees, rule induction, clustering, logistic regression and feature construction algorithms. In the following paragraphs we introduce some basic characteristics of the mentioned algorithms. More details about the motivation of using these algorithms will be provided in the next chapters, when their particular utilization will be discussed.

Decision trees

Decision tree learning is one of the most used methods for inductive inference. It is a classification method that approximates a discrete-valued target function.

Decision trees are constructed using only those attributes best able to differentiate the concepts to be learned. The selection of the attribute to be used for splitting is determined by measures as *information gain* or *gain ratio* (Mitchell [1995]). They measure how well a given attribute separates the training examples according to the target classification. A decision tree is built by initially selecting a subset of instances from a training set. This subset is then used by the algorithm to construct a decision tree. The remaining training set instances test the accuracy of the constructed tree. The remaining training set instances test the accuracy of the constructed tree. If the decision tree classifies the instances correctly, the procedure terminates. If an instance is incorrectly classified, the instance is added to the selected subset of training instances and a new tree is constructed. This process continues until a tree

that correctly classifies all nonselected instances is created or the decision tree is built from the entire training set.

To improve the readability, the learned trees can be converted into sets of if-then rules. ID3 (Quinlan [1996]) and C4.5 (Quinlan [1993]) are among the most well-known decision-tree algorithms. The improved version of ID3, C4.5 (and its commercial version C5.0) includes methods for dealing with numeric attributes, missing values, noisy data and generating rules from trees. In this thesis, we will use C4.5 algorithm to characterize the logistic homogeneous clusters developed in Chapter 3 and to induce predictive models.

Rule induction

If-then rules are human readable representations of induced models. They may sometimes be preferable to decision trees, which can result in very complex and difficult trees. One way to obtain a more readable representation is first to learn a decision tree and second, to translate the tree into a set of rules (e.g. C4.5rules (Quinlan [1993])). Another alternative is to employ algorithms that directly learn sets of rules. One approach for rule induction is the *covering* approach, which at each stage identifies a rule that “covers” some of the instances. The covering approach considers each class and seeks a way of covering all instances in it, at the same time excluding instances not in the class. Algorithms for rule induction provide rules in propositional representation, or allow for more expressive rules, e.g. learn rule-sets of first-order rules that contain variables. Some algorithms for rule induction use general to specific beam search, like CN2 (Clark and Niblett [1989]) and AQ (Michalski [1969]).

In our thesis, we use an algorithm called Ripper, that induces rule-sets (Cohen [1995]). It has been shown that Ripper is competitive with C4.5rules in terms of error rates, but more efficient than C4.5rules on noisy data (Cohen [1995]). The method used to induce rule sets is as following: first, all classes are ordered. The ordering is always in increasing order of prevalence. The algorithm is used to find a rule set that separates the least prevalent class from the remaining classes. Next, all instances covered by the learned rule set are removed from the data set, and the algorithm is used to separate the next least prevalent class. This process is repeated until a single class remains, which will be used as the default class.

Clustering

Clustering is considered as a method of unsupervised learning. Unlike classification, the groups are not predefined. The grouping is performed by finding similarities between data. A basic issue in clustering is to define the similarity metric for the considered data. Subsequently, clusters can be defined as follows (Dunham [2003]):

- Set of alike elements. Elements from different clusters are not alike.
- The distance between points in a cluster is smaller than the distance between a point in the cluster and any point outside it.

Clustering algorithms can be categorized as *hierarchical* or *partitional*. In hierarchical clustering, a nested set of clusters is created and the number of clusters is not fixed beforehand. Hierarchical algorithms can be further categorized as *agglomerative*,

i.e., clusters are created in a bottom-up fashion, and *divisive* algorithms that work in a top-down fashion. Hierarchical algorithms differ in the way they compute the distance between items. Well-known hierarchical algorithms are single link, complete link and average link (Dunham [2003]). Partitional clustering results into just one set of clusters, containing a fixed number of clusters.

In this thesis we use a clustering method available in the Clementine 6.0.1 SPSS product (Clementine [SPSS Inc., 2000]), called the 'Two-Step' method. The first step makes a single pass through the data, during which it compresses the raw input data into a manageable set of sub-clusters. The second step uses a hierarchical clustering method to progressively merge the sub-clusters into increasingly larger clusters, without requiring another pass through the data (Clementine [SPSS Inc., 2000]).

Logistic regression technique

Binomial (or binary) logistic regression is a form of regression which is used when the dependent is a dichotomy and the independents are of any type. Logistic regression can be used to predict a dependent variable on the basis of independents and to determine the percent of variance in the dependent variable explained by the independents, to rank the relative importance of independents and also to assess interaction effects.

Logistic regression has many analogies to the linear regression; however, logistic regression does not assume linearity of relationship between the independent variables and the dependent, does not require normally distributed variables, does not assume homoscedasticity, and in general has less stringent requirements. The success of the logistic regression can be assessed by looking at the classification table, showing correct and incorrect classifications of the dependent variable. Also, goodness-of-fit tests are available as indicators of model appropriateness to test the significance of individual independent variables (e.g. Wald statistic).

Feature construction and data reduction

The role of representation has been recognized as a crucial issue in Artificial Intelligence and Machine Learning. In the paradigm of learning from examples and attribute-value representation of input data, the original representation is a vector of *attributes* (features, variables) describing *examples* (instances, objects). The transformation process of input attributes, used in feature construction, can be formulated as follows (Bhanu and Krawiec [to appear]): given the original vector of features and the training set, construct a derived representation that is better given some criteria (i.e., predictive accuracy, size of representation). The new transformed attributes either replace the original attributes or can be added to the description of the examples. Examples of attribute transformations are counting, grouping, interval construction/discretization, scaling, flattening, normalization (of numerical values), clustering, principal component analysis, etc. (van Someren [2001]). Many transformations are possible, by applying all kinds of mathematical formulas, but in practice, only a limited number of transformation are really effective.

In this thesis, we transform the input attributes for two reasons: (i) to operationalize a certain construct and (ii) to reduce the data dimensionality for inducing

a predictive model with a better predictive accuracy. Well-known algorithms for dimensionality data reduction are Principal Component Analysis and Factor Analysis.

In Section 3.4 the aggregation of data by operationalizing the concept of process complexity into specific variables is presented. The raw data are transformed using operations like counting, mean and variance calculation. Performing these types of transformations, we call *data aggregation*. In Section 3.5 we use Principal Component Analysis to reduce the size of our input data. The goal is to find the best model, by comparing the predictive accuracy of the model induced from all input features and with the predictive accuracy of the model induced by using factors resulted from the Principal Component Analysis.

Developing and evaluating machine learning models

As mentioned in (van Someren [2001]), “constructing a machine learning model in general is not a matter of simply applying a technique”. Therefore, for inducing a machine learning model, a certain methodology should be applied. In (Weiss and Indhurkya [1998]) the following general methodology is presented:

1. Select a model class. This involves defining a language for representing models. This language consists of a structure and a vocabulary and an interpretation from a model to the examples.
2. Collect the data.
3. Clean the data by removing examples or values that violate certain constraints and estimating missing values.
4. Transform the data and formulate these so that models can be constructed.
5. Apply an induction method to construct a model.
6. Interpret the model and/or use the model to make predictions about new data.

Finding the method appropriate to the problem is considered to be an art and the same holds for transforming the problem data such that a particular method can be applied (van Someren [2001]).

A very important methodological issue is to insure that a learned model is credible, i.e. to evaluate the model in terms of its performance. For the evaluation, we need two data sets: the *training set* (seen data) to build the model, and the *test set* (unseen data) to measure its performance. Sometimes, we need also a *validation set* to tune the model, for example to prune a decision tree. All three data sets have to be representative samples of the data that the model will be applied to.

The *holdout* method, represents a single train-and-test experiment, typically between 1/3 and 1/10 held out for testing. However, a single random partition can be misleading for small or moderately-sized samples, and multiple train-and-test experiments can do better.

N-fold cross-validation (*n-fold cv*) is another evaluation method that can be used to see how well a model will generalize to new data. The data set is divided into n subsets. Each time, one of the n subsets is used as the test set and the other $n-1$ subsets are put together to form a training set. Then the average error rate across all

n trials is computed. Every data point gets to be in a test set exactly once, and gets to be in a training set $n-1$ times. The variance of the resulting estimate is reduced as n is increased. Typically $n=10$ is used (Weiss and Kulikowski [1991]).

Leave-one-out is an elegant and straightforward technique for estimating classifier performance. Because it is computationally expensive, it has often been reserved for problems where relatively small sample sizes are available. For a given method and sample size, n , a classifier is generated using $(n - 1)$ cases and tested on the single remaining case. This is repeated n times, each time designing a classifier by leaving-one-out. Thus, each case in the sample is used as a test case, and each time nearly all the cases are used to design a classifier. While leaving-one-out is a preferred technique, with large sample sizes it may be computationally quite expensive. However, as the sample size grows, other traditional train-and-test methods improve their accuracy in estimating performance.

When comparing the performance of two machine learning algorithms on the same data set, different performance measure can be used, depending on the problem, i.e., if we have to classify qualitative instances or if we have to predict numerical instances. When predicting a qualitative output, *classification* is used to map examples into predefined groups or classes. When predicting a quantitative output, a *numeric prediction* is performed.

For **classification** problems, we measure the performance of a model using the *error rate*, *precision*, *recall* and *F-measure*.

Error rate represents the percentage of incorrectly classified instances in the data set. Comparing two models, it can be tested if the difference between the error rates is significant by doing a paired t-test (StatSoft [2000]). The error rate in itself is not very meaningful, therefore we have to compare against a baseline model. The simplest *baseline model* assigns a classification randomly. If the distribution of data is skewed, the frequency baseline can be used, which always assigns the most frequent class. Its error rate is $1 - f_{max}$, where f_{max} is the percentage of instances in the data that belong to the most frequent class.

The error rate is an inadequate measure of the performance of the algorithm because it does not take into account the *cost* of making wrong decisions. For example, if our problem is to detect an oil slick into the sea we can distinguish between *false positive*, i.e. wrongly identifying an oil slick if there is none and *false negative*, fail to identify an oil slick if there is one. In such a situations, false negatives are much more costly because they lead to environmental disasters than false positive, which are only false alarms. Three measures commonly used in information retrieval and machine learning in general, are precision, recall and the harmonic mean of these two, namely the F-measure.

Table 2.1: The confusion matrix

		Predicted class	
		<i>yes</i>	<i>no</i>
Actual class	<i>yes</i>	true positive (TP)	false negative (FN)
	<i>no</i>	false positive (FP)	true negative (TN)

Precision is the number of class members classified correctly over the total number of instances classified as class members (Equation 2.1). *Recall* is the number of class members classified correctly over total number of class members (Equation 2.2). Precision and recall can be combined into the *F-measure*, their harmonic mean (Equation 2.3). For example, in case of the oil slick scenario, we prefer to maximize recall (i.e. to avoid environmental disasters), maximizing precision (i.e. avoiding false alarm) being not so important.

$$Precision = \frac{|TP|}{|TP| + |FP|} \quad (2.1)$$

$$Recall = \frac{|TP|}{|TP| + |FN|} \quad (2.2)$$

$$F = \frac{2 * Recall * Precision}{Recall + Precision} = \frac{2 * |TP|}{2 * |TP| + |FP| + |FN|} \quad (2.3)$$

However, the quality measure offered by these measures are no longer appropriate when numeric quantities have to be predicted: errors are not simply present or absent, they come in different magnitudes. One evaluation measure often used for **numeric prediction** is the *mean square error*, that measures the mean difference between actual and predicted values (Equation 2.4).

$$MSE = \frac{1}{n} * \sum_{i=1}^n (a_i - p_i)^2 \quad (2.4)$$

where p_1, \dots, p_n are the predicted values for the instances $1, \dots, n$ and a_1, \dots, a_n the actual values of the instances $1, \dots, n$. Other evaluation measures used in case of evaluating numeric predictions are mean absolute error, relative squared error, correlation coefficient, etc. (Witten and Eibe [2000]).

2.3 Research design

The research design is presented in Figure 2.1. Our claim is that process data can reveal useful knowledge about the investigated process. The available data should be manipulated in such a way as to provide as many insights as possible about the investigated process. We want to demonstrate that data can be useful if represented as aggregated data and sequence data, which is expressed by two research perspectives, described in Part II and Part III.

In Part II the first research perspective of this thesis is presented, on the usefulness of aggregated data in understanding processes. In Chapter 3, the concept of process complexity have been operationalized into specific variables. Immediately following aggregation, logistic homogeneous clusters are developed. Knowing some case characteristics (patient characteristics such as age, gender, chronic diagnosis) and the patient groups, a rule-set is induced for coordinating patients inside the hospital, i.e., the future route of patients can be predicted.

In Part III the second research perspective of this thesis is presented, on the use of sequence data. Sequence data constitute information recorded in process logs,

recording the process steps that have been executed over time. The goal is to derive a process model explaining the events recorded.

In practical situations it seems realistic to assume that process logs contain noise. Noise can have different causes, such as missing registration data or input errors. Moreover, the log can be incomplete if not all possible sequences of tasks appear in the log.

Two different approaches are considered, depending on the recorded log information. When there is no noise in the process log and there is “sufficient” information, a process model can be built, as discussed in Chapter 4. Moreover, it is possible to identify the class of models for which it is possible to rediscover the model accurately, by looking to the process log. However, real-world process logs are often noisy and incomplete. The alternative is to use statistical and machine learning techniques to induce decision models that can be used to construct the process model, as shown in Chapter 5. In both situations, the final goal is to construct a process model that can be further analyzed and eventually optimized.

We evaluate our research perspective via some applications, discussed in Chapter 6. The applications are considered from different domains and are intended to show that our proposed method can be generalized and that it is able to result in useful insights into the considered process.

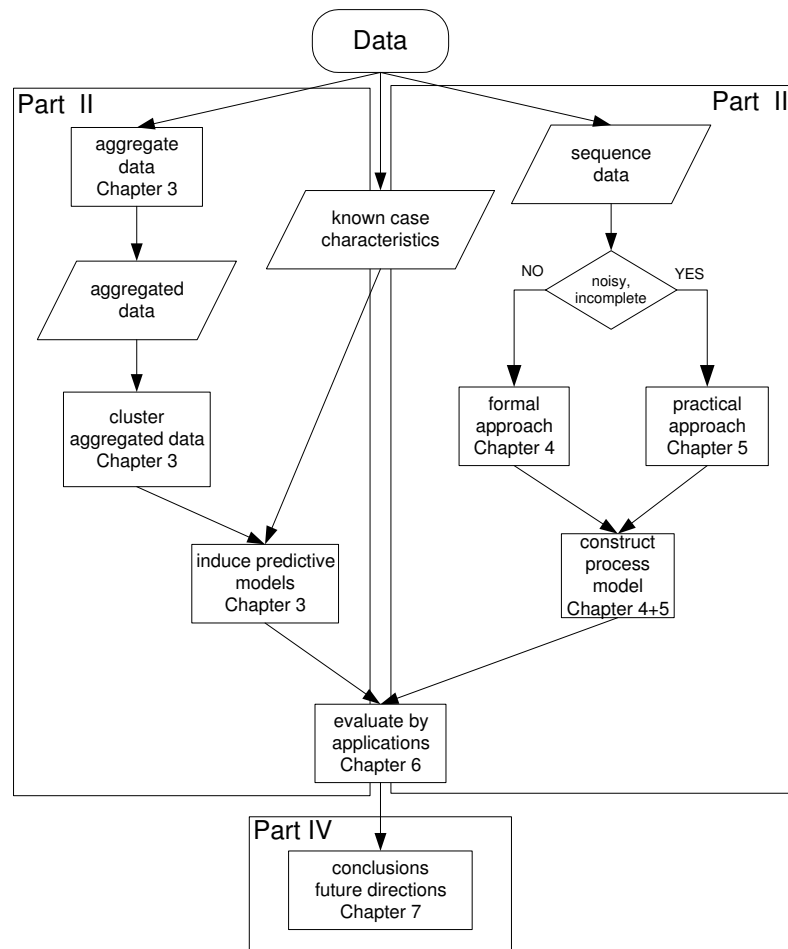


Figure 2.1: The research design.

Part II

Process modelling via clustering aggregated measures

Chapter 3

Modelling the process of multi-disciplinary patients

In this chapter we elaborate on the models that can be induced by machine learning on the basis of aggregated data ¹. We describe how raw data are aggregated by operationalizing the logistic complexity. In the medical domain, the logistic patient-care process can be characterized by constructing homogeneous logistic groups. Based on some known a-priori patient information (such as age, gender, chronic diagnosis) and the logistic groups, we induce a rule-based predictive model that assigns a patient to a logistic cluster, in order to predict the future route of patients inside the hospital.

In doing so, we aim to respond to the second and third research questions mentioned in Chapter 1, i.e.:

2. How can machine learning techniques be used for the clustering of process-related measures?
3. Knowing that relevant clusters can be developed, how can they be used to make predictions?

The structure of this chapter is as follows: in Section 3.1 we introduce the problem of modelling the process of multi-disciplinary patients. In Section 3.2 the medical and logistic domains are briefly discussed. We provide a medically-oriented description of the multi-disciplinary patients, all treated for Peripheral Arterial Vascular (PAV) diseases. From the logistic point of view, we then elaborate on the importance of the underlying processes of medical multi-disciplinary patients, particularly when one aims to optimize the patient throughput. In Section 3.3 we describe the collection and preparation of data. The aggregation of the raw data for operationalizing the logistic complexity is presented in Section 3.4. Section 3.5 describes the clustering experiments for finding logistically homogeneous groups of patients. Our approach to develop predictive models is presented in Section 3.6. In Section 3.7 we discuss the results of the data mining techniques used. Finally, in Section 3.8 we formulate conclusions on the basis of our current findings.

¹The content of this chapter is based on the work that has been appeared in (Mărușter et al. [2002c]).

3.1 Introduction

In the Netherlands, as in many other countries in the world, there is a markedly growing demand for the coordination of patient care. Strong emphasis is placed on medical and organizational efficiency and effectiveness to control national health care expenditures. One of the recognized efficiency problems is that sub-optimally coordinated care often results in redundant and overlapping diagnostic procedures performed by medical specialists from different specialties within the same hospital. Coordination becomes especially important when hospitals structure their health care into specialty-oriented units, and care for patients is not constrained within single units. From a logistic point of view, this creates a tension between, on the one hand, the control over the units, and on the other hand, the coordination needed among units to control the patient flow. The total flow of patients in a hospital can be divided into mono-disciplinary patients and multi-disciplinary patients. Multi-disciplinary patients require the involvement of different specialties for their medical treatment. Naturally, these patients require more efforts regarding the coordination of care. A possible solution is the creation of new multi-disciplinary units, in which different specialties coordinate the treatment of specific groups of patients. A first step in this solution is to identify salient patients groups in need of multi-disciplinary care. Furthermore, adequate selection criteria must exist to select new patients for treatment in a multi-disciplinary unit. As we will demonstrate, grouping and classification techniques seem to offer a solution.

In the medical domain, various grouping and classification techniques are developed and used (Casemix [2001], Fetter [1983]). They can be categorized by their purposes as utilization, reimbursement, quality assurance and management applications (Ploman [1985]). For example, Fetter's Diagnostic Related Groups (DRG) (Fetter [1983]) and their refinements (Fetter and Averill [1984]) are homogeneous in terms of use of resources, but the elements within a single group show rather high variability and low homogeneity from the underlying process point of view (de Vries et al. [1998b]). Starting from the original DRG concept, researchers and professionals organized themselves into a joint network for providing efficient methods for health management at different levels of care, under the name of case-mix classification systems (Casemix [2001]). However, none of the existing classification systems are homogeneous from the underlying logistic process point of view (de Vries et al. [1998b]). A solution will be to consider a logistic classification system that results in a higher logistic homogeneity of groups.

We investigate the possibility of building an alternative, logistic-driven grouping and classification system for medical multi-disciplinary patients with the aid of machine learning techniques. In the medical domain, machine learning methods are used successfully for diagnostic, prognostic, screening monitoring, therapy support purposes (Kononenko [2001], Lavrač [1993]), but also for overall patient management tasks such as planning and scheduling (Miksch [1999], Spyropoulos [2000]).

In Section 3.4 we aggregate raw data in order to operationalize the concept of logistic complexity of a process. In Section 3.5 the aggregated data are used for developing logistically homogeneous groups within the population of patients with PAV (Peripheral Arterial Vascular) diseases.

Table 3.1: Patients with PAV diseases expressed in medical terms

Pathologies	Intermediate stage	Manifestation	Measurable and visible symptoms/complaints	Irreversible disorders and diseases
Arteriosclerosis	Plaque thrombus	Ischaemia	Pain in legs	Impair of organs, muscles and arteries
Disturbed composition of the blood	Plaque thrombus	Ischaemia	Pain in chest	Impair of organs, muscles and arteries
Disturbed metabolism	High concentration of glucose in blood	Insufficient supply of glucose in cells	Fatigued, perspiration, tremble	Disorder of arteries affection of nerves

3.2 Understanding the problem domain

3.2.1 The medical problem

Patients who require the involvement of different specialties are hardly a new phenomenon in health care. In general, one can say that this group of patients is increasing because of the increasing specialization of doctors within the hospital and an aging population. Recent studies in the Netherlands show that approximately 65% of the patients visiting a hospital are multi-disciplinary (de Vries et al. [1998a]). Consequently, certain special arrangements have emerged for these patients. For instance, some hospitals have special centers in which different specialties work together on backbone problems.

Patients with peripheral arterial vascular (PAV) diseases (peripheral refers to the entire vascular system except for the heart and brains) are a good example of multi-disciplinary patients. Surgery, internal medicine, dermatology, neurology and cardiology are the specialties most frequented involved by the treatment of these patients. Alarmingly, a recent study of the Netherlands Heart Foundation shows that the care for these patients leaves much to be desired, because it is too dispersed: it is difficult for doctors in primary health care to know what specialty to refer to; knowledge within the hospital is dispersed; there is a lack of within-hospital co-operation; and there are impediments to scientific research.

Arguably, one important reason for these problems is that patients with PAV diseases are grouped on the basis of medical homogeneity, in the hope that this will result in logistically homogenous groups. However, PAV are a variety of diseases, both acute and chronic, life-threatening, or invalidating. Table 3.1 illustrates that describing these diseases as a group is complex. One complaint can have many different causes, one cause can have different manifestations and there is complexity in cause and effect between pathologies.

One of the consequences of the complexity of expressing these patients in medical terms is that the homogeneity of the underlying treatment processes of these patients is low. This leads us to the logistic perspective of our approach.

3.2.2 The logistic problem domain

Logistics is defined as “the coordination of supply, production and distribution process in manufacturing systems to achieve a specific delivery flexibility and delivery reliability at minimum costs” (Bertrand et al. [1990], de Vries et al. [1999]). Translated to health care organizations, it comprises the design, planning, implementation and control of coordination mechanisms between patient flows and diagnostic and therapeutic activities in health service organizations. The goal is to maximize output/throughput with available resources, taking into account different requirements for delivery flexibility (e.g., differentiating between elective/appointment, semi-urgent, and urgent delivery) and acceptable standards for delivery reliability (e.g., determining limits on waiting list length and waiting times) and acceptable medical outcomes (de Vries et al. [1998a], Vissers [1994]).

First of all, a production control approach to hospitals requires knowledge about processes. However, the main characteristic of hospital products is that they are organized by specialty: internal medicine, cardiology, pulmonology, etc. The physicians belonging to a specialty are specialized in treating complaints in a well-defined part of the human body; often there are even sub-specializations within a specialty, for instance diabetics, enterology and oncology as specializations within internal medicine. However, from a logistic point of view we are looking for homogeneity of the underlying processes. With this we mean the sequence, timing and execution of activities for patients by the hospital staff (specialists, nurses and paramedics). Distinguishing logistically homogeneous groups appears to be important, because every logistic group can require its own optimal control system. Subsequently, in the following sections we will investigate whether such logistic groups can be found in reality.

3.3 Data collection and preparation

The two logistic characteristics to typify a production situation, or in this case, the care process of a patient, are (i) the complexity of the care process and (ii) the routing variability of the care process. In this research, we concentrate on the first type, i.e. the complexity of the care process of a patient. Keep in mind, we are referring to logistic complexity, which can be something completely different from medical complexity. Before we come to the part of operationalising the characteristics, a remark on the subject of data gathering in hospitals is in place. The degree of detail in the majority of hospital registrations is high, but the information is normally hidden in different databases. Relevant patient information can be found in clinical databases, outpatient databases, laboratory databases, etc. When all patient information is gathered, we have the hospital history of the patient.

When planning to do quantitative investigations based on real data, one has to be aware that “some critical steps should be followed” (Dilts et al. [1995]). What we plan to do is cluster analysis in order to find the logistically homogeneous groups. Therefore, we concentrate on the following aspects: (i) data selection, (ii) the attributes (or variables) that should be recorded (measured) and (iii) how to deal with missing data.

3.3.1 Data selection

The first step for data selection is to establish the criteria on which the data will be chosen. Our target is to investigate PAV patients, because they are a good example of complex multi-disciplinary patients. Therefore, we need to specify what we mean with a PAV patient. For this purpose, interviews were held with specialists from the source hospitals, which revealed that certain types of diagnoses point to our PAV patients. These diagnosis resulted in three lists: (a) a list with degenerative underlying chronic diseases, (b) a list with PAV diseases, and (c) a list with diagnoses related with chronic or PAV diseases. We selected the whole population of patients who have at least one diagnosis from list (a) or (b) from the Elisabeth Hospital located in Tilburg, the Netherlands. Note that we work with the complete population of PAV patients, and not with a sample. The degenerative underlying chronic diseases from list (a) (e.g. diabetes) are the cause of a lot of PAV diseases, therefore together with diagnoses from list (b) they have been considered as selection criteria. For patients with diagnoses from list (a) or (b), all records related to visits in different departments of the hospital were extracted. These records contain information mainly related with:

- personal characteristics: age, gender, address, date of birth and date of death if the patient is deceased, etc.
- characteristics of the polyclinic visit: specialist, date, referral date, referring specialist (if the general practitioner requests the visit or another specialist from the hospital), urgency, etc.
- characteristics of the clinical admission: specialist, date, diagnosis (1 main diagnosis and up to 8 possible secondary diagnoses), treatment, referring specialist (if the general practitioner requests the admission or another specialist from the hospital), urgency or planned admission, etc.
- radiology, functional investigations information, other investigations.

These information fields were used to build a time-ordered history for 3603 patients. Please note that our purpose is not to analyse the underlying processes in the patient's history. For instance, given a patient who breaks a leg in February, and undergoes an appendectomy in August, we find both events in the patient's history, but we do not want to consider the two facts as one medical case. To this end we established, with the aid of medical specialists, a set of heuristic rules for splitting the patient's history into separate medical cases. We considered only those medical cases that contain at least one clinical admission (because only in case of clinical admission we have recorded the diagnosis). The end result was a database with 4395 records as medical cases of the 3603 considered patients.

3.3.2 Missing data

The existence of missing data should be carefully investigated in case of performing clustering analysis, because the possible missing data should be replaced with some estimates (Dilts et al. [1995]). However, in our case we plan to cluster aggregated variables. The aggregation method chosen to operationalise the logistic complexity

into logistic aggregate variables is filtering out missing values and the aggregated variables themselves do not have missing values. It can be safely expected that possible missing data that exist in the medical case log (our raw material) will not significantly affect the clustering results.

3.4 The aggregation of raw data

The literature does not offer a unique measurement of care process complexity. Based on existing logistic literature concerning complexity, we operationalize the concept of complexity of the underlying process by distinguishing six aggregated logistic variables, each to be investigated as a potential (partial) measurement of care process complexity. We build from the raw data six aggregated logistic variables as described below. To illustrate the construction of the logistic variables, we used the following abbreviations: “I” represents Internal medicine, “C” represents Cardiology and “D” represents Dermatology.

1. C_{dif_visit} : the total count (number) of involved specialties within the medical case. The assumption is that the more specialties are involved, the more complex the medical case is. Suppose that a medical case contains a sequence of visited specialties as follows: I-I-C-I-D-I. Thus, the logistic variable $C_{dif_visit} = 3$.
2. C_{shift} : the number of shifts within the medical case, counted by the total number of visits to specialties within the medical case. The assumption is that the more a patient has to go from one specialty to another, counted by the total number of visits, the more complex the medical case. As an illustration, let us consider the following example. Consider that patient A has a medical case that involves the following sequence of visited specialties: I-I-C-I-D-I; C_{shift} will be computed as the number of shifts divided with the total number of visits, within the medical case, i.e. $C_{shift_A} = 4/6 = 0.6$. Consider now that patient B has a medical case where the specialties are in the sequence I-I-C-I-I-I-D-I-I-I-I. Thus, $C_{shift_B} = 4/13=0.3$. Obviously, patient A is more complex than patient B, although both A and B “changed” specialties four times. Thus, the more a patient has to go from one specialty to another, counted by the total number of visits within the medical case, the more complex the medical case.
3. N_{visit_mc} : number of visits within the medical case per time-scale. The assumption is that the more visits per time-scale, the more complex the medical case. For example, consider that patient A visited three specialties in four weeks, whereas patient B visited three specialties in twelve weeks. Subsequently, $N_{visit_mc_A} = 3/4 = 0.7$ and $N_{visit_mc_B} = 3/12 = 0.2$, consequently patient A is more complex than patient B.
4. N_{shift_mc} : number of shifts within the medical case per time-scale, counted by the total number of visits to specialties. The assumption is that the more shifts per time-scale, the more complex the medical case. For example, consider that patient A has a medical case that involves the following sequence of visited specialties in four weeks: I-I-C-I-D-I. Patient B visited the following specialties in twelve weeks: I-I-C-I-I-I-D-I-I-I-I. Hence, $N_{shift_mc_A} = 0.6/4 = 0.15$,

Table 3.2: Example of visited specialties in four months (January, February, March and April) for patient A and B and the correspondent mean and variance

Patient	Jan.	Feb.	March	April	Mean	Variance
A	I-I-D	I-I-I	-	I-D-C	Mean(1/3, 0, 2/3)=0.3	Var (1/3, 0, 2/3) = 0.11
B	I-I-I	C-I	I-I	I-I-D-I-I	Mean (0,1/2,0,2/5)=0.2	Var (0,1/2,0,2/5) = 0.06

$N_shift_mc_B = 0.3/12=0.025$ and consequently, patient A is more complex than patient B.

5. M_shift_mth : mean of number of shifts (counted by the total number of visits to specialties) per month. Within a medical case, for each month the number of shifts (by the total number of visits to specialties) is calculated, next the mean is computed. The higher the mean, the higher the complexity of the medical case. Suppose that patients A and B have the sequences of visited specialties in the months January, February, March and April as shown in Table 3.2. Because $M_shift_mth_A = 0.3$ and $M_shift_mth_B = 0.2$, patient A is more complex than patient B.
6. Var_shift_mth : variance of number of shifts (counted by the total number of visits to specialties) per month. Within a medical case, for each month the number of shifts (counted by the total number of visits to specialties) is calculated, next the variance is computed. The higher the variance, the higher the complexity of the medical case. As we can see from Table 3.2, patient A is more complex than patient B.

In the next section we see how the six aggregated variables described above are used for developing logistically homogeneous groups within the population of patients with PAV diseases. Using a-priori patient personal information, the predictive model is induced to assign a patient that just started its treatment process to the most suitable logistic group.

3.5 Development of logistic patient groups

We plan to use clustering techniques to group multi-disciplinary patients in homogeneous logistic groups. If relevant clusters of patients can be found, these groups can be used in two ways: (i) to predict as early as possible to what cluster an individual patient belongs and (ii) to develop different logistic control systems for each homogeneous group. In this research, we concentrate only on the first usage, namely to predict the cluster to which a patient is likely to be assigned.

We combine unsupervised and supervised machine learning techniques to achieve our twofold objective:

1. First, we want to classify patients in groups that are homogeneous from the underlying process point of view. For this purpose, we already operationalized the concept of logistic complexity into different aggregate logistic variables that will

be used further in clustering. Subsequently, we will characterize the obtained clusters by inducing rules based on the aggregated logistic variables.

2. Second, we aim at developing a rule-based predictive model that can assign a new patient on the basis of some given personal information (age, gender, chronic diagnosis), to the most suitable logistic group instantly. Thus, the a-posteriori information encapsulated in the aggregated logistic variables will be used for the development of homogeneous logistic clusters; conversely, a-priori personal information will be used to assign new patient as soon as possible to a cluster.

We plan to assess the quality of the logistic clusters considering a combination of different criteria. First, we want our obtained clusters to be logistically homogeneous. Second, both the cluster characterization rules and the predictive rules should make sense from the medical point of view; thus we are interested in the *intelligibility* and *usefulness* of our rules.

After the selection and preparation of the data, our next step is the clustering of our patients with PAV diseases into homogenous groups, from the logistic point of view. These homogeneous groups can be characterized by rules that draw on the aggregated logistic variables.

3.5.1 Clustering experiments

Clustering techniques are used to group data into groups that are not known beforehand. As clustering method we chose the Two-Step method, available in the Clementine 6.0.1 SPSS software (Clementine [SPSS Inc., 2000]). The goal of this clustering technique is to (i) minimize variability within clusters and (ii) maximize variability between clusters. The first step makes a single pass through the data, during which it compresses the raw input data into a manageable set of sub-clusters. The second step uses a hierarchical clustering method to progressively merge the sub-clusters into increasingly larger clusters, without requiring another pass through the data.

We chose this type of clustering technique because it shows two types of advantages. First, it is not necessary to decide beforehand the numbers of clusters. Second, compared to other techniques, it is relatively fast for large data sets and large numbers of variables.

For building the logistic patient groups, we ran two series of experiments: clustering experiments based on (i) all six logistic variables built so far, and (ii) factors extracted from the initial six logistic variables. For the latter set of experiments we use factors extracted with the Principal Component Analysis technique, available also in the Clementine software.

3.5.2 Clustering experiment involving all logistic variables

In our first clustering experiment all logistic variables are used. We let the Two-Step method search the number of clusters automatically. The results are given in Table 3.3.

The Two-Step method resulted in three clusters, with 2330, 127 and 1938 items, respectively. In order to choose the valid homogeneous clusters, we compare the

Table 3.3: Means and standard deviations for logistic variables in case of not clustered data (Total) and for the clustering model LOG_VAR_3

Logistic variables	Total	Clustering model LOG_VAR_3		
		Cluster-1 (2330 ^a)	Cluster-2 (127 ^a)	Cluster-3 (1938 ^a)
<i>C_dif_visit</i>				
Mean	3.51	2.566	3.976	4.608
S.D.	1.58	0.758	2.419	1.515
<i>C_shift</i>				
Mean	0.243	0.092	0.43	0.202
S.D.	0.217	0.139	0.215	0.138
<i>N_visit_mc</i>				
Mean	0.085	0.046	1.373	0.048
S.D.	0.286	0.074	0.997	0.045
<i>N_shift_mc</i>				
Mean	0.002	0.0	0.063	0.002
S.D.	0.026	0.002	0.142	0.004
<i>M_shift_mth</i>				
Mean	0.087	0.013	0.077	0.177
S.D.	0.113	0.025	0.13	0.112
<i>Var_shift_mth</i>				
Mean	0.029	0.005	0.006	0.06
S.D.	0.038	0.011	0.012	0.039

^a Number of items in each cluster.

standard deviation of each cluster with the standard deviation of the data not yet clustered. Cluster-1 and cluster-3 seem to show generally higher degrees of homogeneity compared to unclustered data. If we look, for example, in Table 3.3 at standard deviation values for variable *C_dif_visit*, both cluster-1 and cluster-3 have lower values than Total ($0.758 < 1.58$ and $1.515 < 1.58$). In the following analyse we therefore concentrate on cluster-1 and cluster-3.

We identified that it was possible to build two reliable clusters. But how can we interpret them? Different methods are available to characterize the clusters found by a clustering technique. One way to look at them is to investigate their means. However, in this thesis we choose to use Quinlan’s induction algorithm C4.5rules (Quinlan [1993]) to characterize the clusters. Seven rules are induced to characterize cluster-1 and 12 rules for cluster-3. Examples of the induced rules are given in Table 3.4 (for the whole rule set, see Appendix B). For each rule we have information about its coverage and the reliability. For instance, if we look at Rule #1 for cluster-1, there are 1943 examples covered by the IF-part of this rule, and 99.9 of them actually belong to cluster-1.

Inspecting the induced rules, the two clusters can be characterized as follows: cluster-1 includes “moderately complex” PAV patients, while cluster-3 covers the “complex” examples. As general characteristics, patients from the “moderately com-

Table 3.4: Some examples of the rules that characterize the different clusters based on all logistic variables.

Rule number	Rule description	Coverage	Reliability(%)
Rule#1, cluster-1	IF $C_dif_spm \leq 3$ and $C_shift \leq 0.296$ and $N_visit_mc \leq 0.506$ and $M_shift_mth \leq 0.101$ and $Var_shift_mth \leq 0.042$ THEN cluster-1	1943	99.9
Rule #11, cluster-3	IF $C_dif_spm > 3$ and $C_shift > 0.304$ and $N_visit_mc \leq 0.688$ THEN cluster-3	1303	97.9

Table 3.5: Some examples of rules that characterize cluster-2 of the clustering model based on all logistic variables.

Rule number	Rule description	Coverage	Reliability(%)
Rule#1, cluster-2	IF $N_visit_mc > 0.688$ THEN cluster-2	87	97.8
Rule #11, cluster-2	IF $C_dif_spm \leq 6$ and $N_visit_mc > 0.506$ and $M_shift_mth > 0.074$ THEN cluster-3	22	91.7

plex” cluster have visited up to three different specialists and show lower values for the shift characteristics, while patients from cluster “complex” have visited more than three different specialists and the values for shift features are higher. Cluster-2 seems to contain the 127 cases that cannot be grouped in cluster-1 or cluster-3. Two interesting rules (displayed in Table 3.5) are induced to characterize this rest cluster.

The patients in cluster-2 show a higher number of visits counted by the duration of the medical case (variable N_visit_mc) than patients from cluster-1 and cluster-3, while the number of different specialists C_dif_spm is not so high. These rules give rise to the impression that patients who repeatedly visit one specialist are in this cluster. Inspection of the data reveals that these patients frequent the dialysis department. Because this is not a PAV-related cluster, we excluded this cluster from our further analysis.

3.5.3 Clustering experiment involving two latent factors

In the previous subsection, we applied the clustering technique directly to the six logistic variables. In this section we first use the Principal Component Analysis extraction method (available in SPSS software, SPSS [SPSS Inc., 2000]) to check

for possible latent factors. We then apply our clustering technique on these latent factors.

We use the Kaiser criterion for retaining factors, i.e., we extract factors with Principal Component Analysis method if the Eigenvalues are exceeding the value 1. A drawback of this criterion is that it can retain too many factors (StatSoft [2000]). In our case, two factors seems to be a reasonable number of extracted factors. In Figure 3.1 is shown the component plot of the results of the Principal Component Analysis.

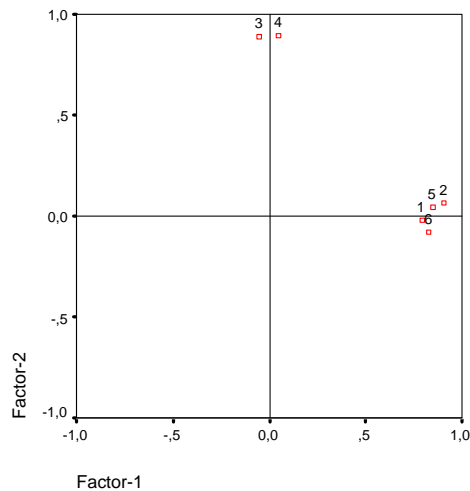


Figure 3.1: Component plot of Factor-1 and Factor-2. The logistic variables are represented by numbers: 1 - *C_dif_spm*, 2 - *C_shift*, 3 - *N_visits_mc*, 4 - *N_shift_mc*, 5 - *M_shift_mth*, 6 - *Var_shift_mth*.

The total variance explained by this model is 74%. Inspecting the two extracted factors (see Figure 3.1 and also Table B.1 from Appendix B), the first factor can be observed showing high correlations with logistic variables *C_shift* (marked with number “2” in Figure 3.1), *M_shift_mth* (marked as “5”), *Var_shift_mth* (marked as “6”) and *C_dif_spm* (marked as “1”), and very small correlations with the rest. The second factor show a high correlation with *N_visits_mc* (marked as “3”) and *N_shift_mc* (marked as “4”) and a low correlation with the other variables.

The factors are difficult to interpret; a hypothesis could be that these two factors represent two facets of complexity. Factor-1 represents somehow the “complexity due to shifts” and Factor-2 “complexity in time span”. Thus, we can conclude that it is worthwhile to search for clusters based on these two factors. In Table 3.6, the mean and standard deviation for the clustering model based on the extracted factors, called *FACTOR.3*, are shown.

Similar to the previous experiments, by comparing the standard deviation of cluster-1 and cluster-3 with the standard deviation of data not yet clustered, cluster-1 and cluster-3 appear to have a higher homogeneity than the unclustered data. The

Table 3.6: Means and standard deviations for the two extracted latent factors, in case of not clustered data and in case of clustering model *FACTOR_3* with three clusters.

	Total	Clustering model <i>FACTOR_3</i>		
		Cluster-1(2936 ^a)	Cluster-2(154 ^a)	Cluster-3(1305 ^a)
<i>Factor – 1</i>				
Mean	0	0.552	0.202	1.267
S.D.	1	0.547	0.81	0.565
<i>Factor – 2</i>				
Mean	0	0.133	3.266	0.087
S.D.	1	0.136	4.11	0.163

^aNumber of items in each cluster

Table 3.7: Some examples of rules that characterize the different clusters based on two latent factors.

Rule number	Rule description	Coverage	Reliability(%)
Rule#1, cluster-1	IF $C_dif_spm \leq 4$ and $C_shift \leq 0.467$ and $N_visit_mc \leq 0.492$ and $M_shift_mth \leq 0.086$ and $Var_shift_mth \leq 0.03$ THEN cluster-1	2165	100
Rule #4, cluster-2	IF $N_visit_mc > 0.604$ THEN cluster-2	97	85.9
Rule #1 cluster-3	IF $C_dif_spm > 4$ and $C_shift > 0.32$ and $Var_shift_mth > 0.027$ THEN cluster-3	716	99.9

values of Factor-1 and Factor-2 for standard deviation and for the mean are 1 respectively 0 in case of unclustered data (Total column in Table 3.6), because the Principal Component Analysis extracts latent factors by standardizing the values of the input variables.

Again, we choose to use Quinlan’s C4.5rules induction algorithm to characterize the clusters. 12 rules are found for cluster-1, with confidences over 85% and 16 for cluster-3, with confidences over 75%. Inspecting the selected rules from Table 3.7 (for the whole rule set, see Appendix B), we arrive at the similar conclusions: there is a cluster for “moderately complex” PAV patients and one for “complex” ones.

The rules look relatively similar, although there are some differences: (i) more rules are based on factors and (ii) for each cluster, there is one rule with a very low coverage and also low confidence; we can interpret it as two rules which try to explain few cases which behave as exceptions. If we remove the two rules for “exceptional” cases for each cluster, we end up with 11 rules for cluster-1 and 15 rules for cluster-

3, with confidences over 93% and 83%, respectively. Moreover, these clusters can be characterized by rules on which basis one cluster contains “moderately complex” PAV patients, and another one “complex” PAV patients, complexity being understood from the logistic point of view. The third cluster contains patients not especially suitable for our purposes: their logistic behaviour is again determined only secondarily by PAV diseases.

The conclusion is that in both situations, (i) clustering based on all logistic variables and (ii) clustering based on two extracted logistic variables, we can obtain homogeneous logistic clusters. The question that we are trying to answer further is: can we use these clusters for prediction purposes? In the next section we compare the two clusters for their capabilities to predict to which cluster a new individual patient belongs.

3.6 Development of predictive models

In the previous section we saw that both clustering methods result in logistic homogeneous clusters. However, if it is not possible to predict to which cluster a new individual patient belongs, the clustering is of little use, except for inspection and interpretation by experts. In this section we investigate if it is possible to use some a-priori personal patient information such as age, gender and previous diagnoses, to predict what kind of logistic behaviour a patient newly entered in the process will have.

Apart from age and gender, a representation of the patient must be generated on the basis of his or her medical history, in order to be assigned to a particular cluster. Knowing to which cluster a patient is likely to belong may provide immediate indications on how to plan future activities, capacity planning, etc. In the following, we describe how we develop predictive models that can be used to assign PAV patients to a certain logistic cluster, based on a-priori information.

A-priori information includes age, gender, primary diagnosis, and potential secondary diagnoses. Age and gender are known for the first time when a patient is registered in the hospital and he/she receives a registration card. Primary diagnoses and potential secondary diagnoses are known only when the patient is clinically admitted. When a patient has a clinical admission, one mandatory primary diagnosis will be recorded and up to eight possible secondary diagnoses. For example, a patient can be admitted in the hospital because of acute gangrene as primary diagnosis; in the same time, this person has a chronic disease, namely arteriosclerosis as secondary diagnosis.

To develop a predictive model that, based on a-priori information, will assign a patient to the most suitable logistic cluster, we use again the C4.5rules algorithm (Quinlan [1993]). The learning material is our database with 4395 medical cases, where the input attributes are age, gender and diagnosis. From the previous clustering phase, we already know for each record (i.e. medical case) to which cluster it belongs, thus each medical case is labelled as “complex” or “moderately complex”. Note that it is possible for a patient to have one medical case that is “moderate complex” and another medical case that is “complex”. In other words, the learning material is composed from records representing histories of medical cases rather than histories of

patients.

Two series of learning experiments were performed for each clustering model, i.e. for the model based on all logistic variables *LOG_VAR_3* and for the model based on two latent factors, *FACTOR_3*. In the first experiment, we want to see if all diagnoses taken as separate input features can result into qualitative predictive rules. In the second experiment, we are interested to test if relevant groups of diseases, e.g. specific chronic diseases and/or heart family of diseases can also provide qualitatively predictive rules. The reason is that, as soon as a specific diagnosis or a group of diagnoses are known, a prediction can be made.

1. Experiment “all diagnoses” with 60 input features: age, gender, total number of diagnoses and 57 possible diagnoses. Each diagnosis is represented as a separate binary feature; if a certain diagnosis is present in the medical case, the corresponding feature is marked with a “1” and with “0” if it is not present.
2. Experiment “chronic diagnoses” with 11 input features: age, gender, total number of diagnoses and 8 diagnosis classes. For this experiment, we created eight diagnosis classes², in which we included all chronic diagnoses: (1) diabetes, (2) hypertension, (3) arteriosclerosis, (4) hyperhomocysteinemia, (5) hyperlipidaemia (including hypercholesterolaemia), (6) coagulation disorders, (7) heart problems and (8) (chronic) renal failure.

3.6.1 Experiment “all diagnoses”

In this first type of experiment we consider 60 input features: age, gender, total number of diagnoses and 57 diagnoses. Each of the 57 diagnoses is taken as a separate feature. Here we are interested to obtain predictive rules in which we can have combinations of age, gender, total number of diagnoses and individual diagnosis. The class (or output) feature is the cluster label, namely “complex” or “moderately complex”. The experiment consists in training and afterwards testing the model, which will result in some rules of a certain quality. The training database contains the following fields:

- Patient ID: number field.
- Age: number field.
- Gender: flag field (1 for male, 2 for female).
- *C_sec_diag*: number field. Represents total number of diagnoses.
- *d****: flag field. This flag will be set to “1” if the patient has the diagnosis coded “***” within the medical case and to “0” if not. For example, if the patient has diabetes, which is coded “250”, the feature *d250* will be marked with “1”.

²These groups of diseases have been indicated by medical specialists.

3.6.2 Experiment “chronic diagnoses”

This second type of experiment consists in 11 input features, namely age, gender, total number of diagnoses and 8 groups of diagnoses. We consider the 6 chronic diagnoses (diabetes, hypertension, arteriosclerosis, hyperhomocysteinemia, hyperlipidaemia, coagulation disorders), heart problems and (chronic) renal failure, each one as separate features. Here we want to test whether specific chronic diseases and/or heart family of diseases can provide qualitatively predictive rules. Also in this type of experiment, the class (or output) feature is the cluster label, namely “complex” or “moderately complex”. The database contains the following fields:

- Patient ID: number field.
- Age: number field.
- Gender: flag field (1 for male, 2 for female).
- *C_sec_diag*: number field. Represents total number of diagnoses.
- g250, g401, g440, ...: flag fields. The diagnoses marked in these fields are all 6 chronic diagnoses. For example, g250 stands for diabetes, g401 for hypertension and g440 for arteriosclerosis. These flags will be set to “1” if the patient has within the medical case that specific diagnosis and to “0” if not.
- heart: flag field. This flag will be set to “1” if the patient has within the medical case at least one diagnosis which relate to heart, and to “0” if not.
- g585: flag field. This flag will be set to “1” if the patient has within the medical case the diagnosis coded 585 (renal failure), and to “0” if not.

We run in total four learning series: one experiment “all diagnoses” with clustering model *LOG_VAR_3*, one experiment “all diagnoses” with clustering model *FACTOR_3*, one experiment “chronic diagnoses” with clustering model *LOG_VAR_3* and one experiment “chronic diagnoses” with clustering model *FACTOR_3*.

The quality of the predictive models is assessed by 10-fold cross-validation (see Section 2.2). The cross-validation performance on test material for experiments “all diagnoses” and “chronic disease” with the two clustering models developed up to now, *LOG_VAR_3* and *FACTOR_3*, is given in Table 3.8.

Since we want to compare the prediction performance of the models that we built so far, we repeat the development of other two alternative clustering models, one based on all logistic variables and one on the two extracted factors. We use the same Two Step clustering method, but we do not let the method find the number of clusters automatically. Rather, we fix the number of final clusters at 2. The resulting model *LOG_VAR_2* consists on two clusters: cluster-1 that contains the same cases (2330) as the “moderately complex” cluster from clustering model *LOG_VAR_3*, and cluster-2 which captures the rest of the cases (2065). In the same manner, model *FACTOR_2* yields two clusters: cluster-1 that contains the same cases (2936) as cluster “moderately complex” from clustering model *FACTOR_3*, and cluster-2 which joins the rest of the cases (1459). The performance of these two models is also shown in Table 3.8.

Table 3.8: Performance of predictive models from experiments “all diagnoses” and “chronic diagnoses”, of clustering models based on all logistic variables (*LOG_VAR_2* and *LOG_VAR_3*) and on two latent factors (*FACTOR_2* and *FACTOR_3*).

Model	No. of elements in each cluster	No. of clusters	Baseline perf.	All diagnosis		Chronic diagnosis	
				Perf.	Gain	Perf.	Gain
<i>LOG_VAR_2</i>	c-1: 2330(53.01%)	2	53	61.2	8.2	63.3	10.3
	c-2: 2065(46.99%)						
<i>FACTOR_2</i>	c-1: 2936(66.80%)	2	67	68.5	1.5	69.4	2.7
	c-2: 1459(33.19%)						
<i>LOG_VAR_3</i>	c-1: 2330(53.01%)	3	53	58.6	5.6	60.5	7.5
	c-2: 127 (2.88%)						
	c-3: 1938(44.09%)						
<i>FACTOR_3</i>	c-1: 2936(66.80%)	3	67	64.1	-	64.6	-
	c-2: 154(3.50%)						
	c-3: 1305(29.69%)						

Of interest are models that show a higher performance than the baseline performance (the percentage of the most common class; in our case, in model *LOG_VAR_2*, cluster-1 comprises 53% of all elements; if the model always predict cluster-1, a performance level of 53% would be attained). As can be seen from Table 3.8, the predictive model with the highest gain in performance concerns the experiment with “chronic diagnoses”, where the cases are labelled based on clusters developed with model *LOG_VAR_2* (all logistic variables and 2 clusters). Its overall performance is 63%; 10% higher than baseline class guessing. The predictive models based on clustering models *LOG_VAR_2* and *LOG_VAR_3* also show a certain gain over the baseline performance. In contrast, the clusters based on the two latent factors show very small gain over the baseline performance, if any.

To illustrate what is learned, we concentrate on the rules of the predictive models from experiment “chronic diagnoses” in case of *LOG_VAR_3*. They are presented in Table 3.9.

The five rules developed for cluster-1 can be shared in two categories: the first three, Rule #1, Rule #2 and Rule #3, which show a low coverage (5, 16 and 6 respectively) and a high confidence (85.7%, 83.3% and 75%) and Rule #4 and Rule #5, with a high coverage (2197 and 3098) and low confidence (62.4% and 61.1%). Because we are interested not only in having high performance (rules with high confidence), but certainty also in wide-coverage general rules that may provide new useful knowledge, we inspect rules Rule #4 and Rule #5 more closely. Using the same reasoning for the rules induced to capture cluster-3, we focus on Rule #2, Rule #3 and Rule #4.

We recall that this experiment type “chronic diagnoses” focuses on a-priori characteristics, i.e. age, gender, total number of diagnoses and 8 groups of diagnoses: diabetes (g250), hypertension (g401), arteriosclerosis (g440), hyperhomocysteinemia (g2704), hyperlipidaemia (g272), coagulation disorders (g286), heart problems (heart) and (chronic) renal failure (g585). The wide-coverage rules tell us that if a patient has three or less diagnoses, and does not have diagnosis g585 (renal failure), it is

Table 3.9: Predictive rules from experiment “chronic diagnoses” with clustering model *LOG_VAR_3*.

Rule number	Rule description	Coverage	Reliability(%)
Rule #1 cluster-1	IF $C_sec_diag > 2$ and $C_sec_diag \leq 3$ and $g250=F$ and $g272=T$ THEN cluster-1	5	85.7
Rule #2 cluster-1	IF Age > 80 and $C_sec_diag \leq 3$ and $g401=T$ THEN cluster-1	16	83.3
Rule #3 cluster-1	IF Age > 91 and $C_sec_diag > 2$ and $C_sec_diag \leq 3$ THEN cluster-1	6	75.0
Rule #4 cluster-1	IF Age ≤ 72 and $C_sec_diag \leq 3$ and $g250=F$ and $g585=F$ THEN cluster-1	2197	62.4
Rule #5 cluster-1	IF $C_sec_diag \leq 2$ and $g585=F$ THEN cluster-1	3098	61.1
Rule #1 cluster-3	IF Age > 65 and Age ≤ 68 and $C_sec_diag > 2$ and $C_sec_diag \leq 3$ and $g272=F$ and $g401=T$ and heart=F THEN cluster-3	11	92.3
Rule #2 cluster-3	IF $g585=T$ THEN cluster-3	93	65.3
Rule #3 cluster-3	IF Age ≤ 72 and Gender=2 and $C_sec_diag > 2$ and $C_sec_diag \leq 3$ and $g272=F$ and $g401=F$ and heart=F and $g585=F$ THEN cluster-3	53	61.8
Rule #4 cluster-3	IF $C_sec_diag > 2$ THEN cluster-3	1259	60.1

likely that he/she will be in cluster-1: a “moderately complex” patient. In contrast, if a patient has diagnosis $g585$ (renal failure), it will be a “complex” patient. Also, according to Rule #4 for cluster-3, if the number of diagnoses is higher than 2, it will be estimated to be a “complex” patient. If the patient does not have diagnosis $g585$,

g401, g272 and heart problems, has in total three diagnoses and is a woman, then she has some chance to be a “complex” patient.

However, the rules provided by this predictive model provide restricted information, regarding only the chronic and heart problems. It may be possible that a patient does not suffer from such diseases, and in such case, as soon as some other individual diseases are known, a prediction can be made. More detailed rules, at the level of individual diagnoses are provided by the predictive model from experiment “all diagnoses” with *LOG.VAR.3*. A selection of the rules with coverage higher than 20 instances and confidence higher than 0.6 is shown in Table 3.10.

Table 3.10: Predictive rules from experiment “all diagnoses” with clustering model *LOG.VAR.3*.

Rule number	Rule description	Coverage	Reliability
Rule #1 cluster-1	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag \leq 2$ and d286=0 and d250=1 and d7802=0 and d440=0 and d4359=0 and d2508=0 and Age > 55 THEN cluster-1	98	61.2
Rule #5 cluster-1	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag \leq 2$ and d286=0 and d250=0 and d425=0 and d997=0 and d446=0 and d413=0 and d428=0 and d426=0 and d441=0 and d443=0 and d707=0 and d2508=0 THEN cluster-1	2383	63.8
Rule #7 cluster-1	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag > 2$ and $C_sec_diag \leq 5$ and d447=0 and d2508=0 and d443=0 and d403=0 and d437=0 and d446=0 and d9972=0 and d357=0 and d250=0 and d426=0 and d410=1 and d4331=0 and d436=0 and d707=0 and d413=0 and d7854=0 and d997=0 and d412=0 and d998=0 THEN cluster-1	51	68.6
Rule #9 cluster-1	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag > 2$ and d447=0 and d2508=0 and d443=0 and d403=0 and	56	62.5

Table 3.10 – continued from previous page

Rule number	Rule description	Coverage	Reliability
	d437=0 and d446=0 and d9972=0 and d357=0 and d250=0 and d426=0 and d410=0 and d427=0 and d459=0 and d5571=0 and d442=0 and d997=0 and d424=0 and d425=0 and d428=0 and d4359=0 and d2720=0 and d413=0 and d412=0 and d444=0 THEN cluster-1		
Rule #1 cluster-3	IF d585=1 and d442=0 and d429=0 and d444=0 THEN cluster-3	74	70.3
Rule #2 cluster-3	IF d585=0 and d2507=1 and $C_sec_diag \leq 7$ and d414=0 and d250=1 THEN cluster-3	53	79.2
Rule #8 cluster-3	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag > 2$ and d447=0 and d2508=1 THEN cluster-3	47	78.7
Rule #13 cluster-3	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag > 3$ and d447=0 and d2508=0 and d443=0 and d403=0 and d437=0 and d446=0 and d9972=0 and d357=0 and d250=0 and d426=0 and d410=0 and d427=1 and d4331=0 THEN cluster-3	47	91.5
Rule #14 cluster-3	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag > 2$ and d447=0 and d2508=0 and d443=0 and d403=0 and d437=0 and d446=0 and d9972=0 and d357=0 and d250=0 and d426=0 and d410=0 and d427=0 and d459=0 and d5571=0 and d442=0 and d997=1 and d412=0 THEN cluster-3	66	62.1

Among the rules induced for cluster-1, we inspect Rule #1: if a patient has diagnosis d250 (diabetes) (and does not have any of the other eight specified diagnoses), has two or less than two diagnoses and an age of more than 55, he or she is likely to be a “moderately complex” patient. Looking at Rule #2 for cluster-3, we can notice that if a patient has in addition to diagnosis d250 the diagnosis d2507 (diabetic foot), he or she will be assigned to cluster-3, which is the cluster for “complex” patients. Subsequently, the number of diagnoses will be higher, which is also according to the rule (number of diagnoses $C_sec_diag \leq 7$). Thus, our model contains a rule that is able to “send” the patient to the right cluster, when an additional diagnosis becomes known. Another meaningful rule is Rule #1 for cluster-3, which says that if a patient has diagnosis d585 (renal failure) and does not have the other three specified diagnoses, he/she will be a “complex” patient. Thus, this rule provides a way to distinguish the patients who have renal failure (and consequently need dialysis), and it can be expected that they will be “complex” patients.

3.7 Discussion

Our first goal was to see whether patients with PAV diseases could be clustered into logistically homogeneous groups. The two different clustering models that we developed, both based on all six logistic variables and two latent factors, show that some reliable clustering is possible. This result can be used as a starting point for building alternative classification models that look for homogeneity from the logistic point of view and not only from the medical point of view.

The two considered approaches, i.e. clustering on logistic variables, and clustering based on latent factors extracted from logistic variables, both lead to three main clusters, of which two hold clear-cut groups of patients: one can be labelled “moderately complex” patients, while the other holds “complex” patients. The remaining third cluster contains a small number of cases that cannot be assimilated to one of the two valid clusters. The rules induced for the characterization of each cluster provide a good insight into the relative importance of the involved logistic dimensions, and here we recall them: (1) C_dif_spm , (2) C_shift , (3) N_visit_mc , (4) M_shift_mth and (5) Var_shift_mth , all these computed per medical case. The rules indicate, for instance, that N_shift_mc may have a low importance: it is never used in any of the rules in the rule set. Tests based on this feature are removed from the rules because they do not contribute enough, apparently, to the classification power of the model. Next to providing information about the logistic variables, the induced rules that distinguish between “complex” patients and “moderately complex” patients can eventually provide reasons for developing a control system.

The grouping models that we develop are fully useful if we are able to combine them with predictive models. Therefore, we are interested to develop predictive models that uses a-priori information to predict in which cluster a patient is likely to be, as soon as the patient enters the health care system. The predictive models obtained so far are rather general. Nevertheless, we can extract some useful information. Look for example to the following rules produced in experiment “all diagnoses” with clustering model LOG_VAR_3 , shown below in Table 3.11.

Rule #1 for cluster-1 says that a patient is “moderately complex” if he/she does

Table 3.11: A selection of predictive rules from experiment “all diagnoses” with clustering model *LOG_VAR.3*.

Rule number	Rule description	Coverage	Reliability(%)
Rule #1 cluster-1	IF d585=0 and d2507=0 and d429=0 and $C_sec_diag \leq 2$ and d286=0 and d250=1 and d7802= and d440=0 and d4359=0 and d2508=0 and Age > 55 THEN cluster-1	98	61.2
Rule #1 cluster-3	IF d585=1 and d442=0 and d429=0 and d444=0 THEN cluster-3	74	70.3
Rule #2 cluster-3	IF d585=0 and d2507=1 and $C_sec_diag \leq 7$ and d414=0 and d250=1 THEN cluster-3	53	79.2

not have diagnosis d585 (renal failure), d2507 (diabetic foot), d429, d286, but has d250 (diabetes) and $C_sec_diag \leq 2$. In contrast, using Rule #2 for cluster-3, a patient is estimated to be “complex” if he/she additionally has diagnosis d2507 (and not diagnosis d585 and d414), increasing the number of diagnoses, i.e. $C_sec_diag \leq 7$. Rule #1 for cluster-3 expresses that as soon as a patient has diagnosis d585 (renal failure), it will be a complex patient (a PAV patient that need dialysis as well). It should be noted that the models presented here are based on a relatively small set of examples, and their outcomes should be taken as indicative of their potential; until there is considerably more data, the obtained predictive rules are not detailed enough and reliable to base a whole control system on.

3.8 Conclusions

In this chapter we showed that raw data can be aggregated by operationalizing the logistic complexity. Also, the induced machine learning models provide useful insights into the treatment process of multi-disciplinary patients.

We proposed a methodology that attempts to offer a solution for a better coordination of patients with peripheral vascular diseases. We shown that by using clustering technique and factor analysis, PAV patients can be shared in two clear-cut clusters, namely “complex” and “moderately complex” patients. These clustering models are relevant if predictive models can be built, based on some known a-priori patient characteristics. Using machine learning techniques, we developed such predictive models and we illustrated that rules can found. The rules that assign patients to clusters also provide clues about which of the six logistic variables that represent a medical case are relevant or not, and in which interaction they are relevant.

In Chapter 6 the developed clusters are further investigated. Namely, the underlying process for each logistic cluster is detected.

Further research should be invested in finding more a-priori patient characteristics that allow predicting logistic clusters more reliably. We plan to do future research by developing a multi-step model. A-priori knowledge as age, gender, risk factors and relevant secondary diagnosis are known the first time a patient enters the hospital. Based on this information, a first prediction could be made and patients could receive the proper treatment faster. Also, when more information becomes available through time (as more steps in the process become known), a secondary more precise prediction can be made. Thus, changes in patient groups and treatments could automatically be discovered and relayed back to the logistic management to inspect whether the new data warrant additional changes in patient groups and treatments.

Part III

Discovering a process from sequence data

Chapter 4

The formal approach

In this chapter we discuss how sequence data can be used in process modelling, assuming that there is no noise in the data and there is sufficient information¹.

We aim to respond to the fourth research question mentioned in Section 1.2 from Chapter 1, i.e.:

4. What kind of processes can be discovered from past process executions?

We present a discovery algorithm that constructs a process model from process logs, using the Petri net formalism. We mention that our own contribution consists of the initial version of this algorithm, illustrated through examples from hospital data in Mărușter et al. [2001, 2002a]. This algorithm has been substantially improved and also a formal analysis has been given by van der Aalst et al. [2003, to appear]². The aim of the analysis was to identify the class of process models for which it is possible to accurately rediscover the model by looking to the process log. This analysis is relevant in connection to the discovery algorithm presented in this chapter, showing its strengths, but also demonstrating its limitations. The idea is to try to overcome these limitations, using a practical approach, as presented in Chapter 5.

The structure of this chapter is as follows: a short introduction into the classical Petri net and workflow net formalisms is provided in Section 4.2. The discovery problem is presented in Section 4.3, describing the algorithm that discovers process models. The analysis of the class of process models for which we can prove that it accurately rediscovers the process model is presented also in Section 4.3. In Section 4.4 we provide an overview of the process discovery literature. We end this chapter with some conclusions in Section 4.5.

4.1 Introduction

Many of today's computerized systems that support business processes (e.g. Enterprise Resource Planning, see Vernadat [1996]) need a process design. The problem is

¹The content of this chapter is based on joint work with Wil van der Aalst, Ton Weijters, Antal van den Bosch and Walter Daelemans, and has appeared in Mărușter et al. [2001, 2002a], van der Aalst et al. [2003, to appear] and van der Aalst et al. [2002].

²In this thesis, we use the formalizations proposed by van der Aalst et al. [2003, to appear].

that a designer has to construct a detailed and accurate model. However, modelling a process requires deep knowledge of the business process at hand, that would imply many discussions with the workers and the management.

Business support systems record different kinds of information for planning, budgeting, bill of materials, distribution and warehousing, flow of work, etc. Due to these systems, much data exist, but unfortunately relevant information is seldom extracted for analysis. Based on the running process, process-related data can be collected and certain analysis can be performed. The result of such an analysis can provide input for (re)designing and re-engineering the business process.

Modelling an existing process is often providing a prescriptive model, that contains what “should” be done, rather than describing the actual process. Subsequently, models tend to be subjective. A modelling method more closer to reality is to use data representing the actual events that took place. The desired outcome is to have process models that are not biased by subjective perceptions or normative behavior.

The idea is to reverse the process and to collect data at runtime, to support process design and analysis. The information collected at runtime, usually recorded in a process log, can be used to derive a model explaining the events recorded. We call this activity *process discovery* (sometimes also referred as *process mining*).

If process activities happen in a way that bypass the system, the process log can still deviate from the actual behavior. Therefore, it is useful to confront man-made models with models discovered by process discovery, to have as deep insight as possible into the process.

In this thesis we choose to represent the discovered process models as Petri nets (Petri [1962]). Because of their good theoretical foundation and the possibility to express concurrency, Petri nets (PN) have been used successfully to model and analyze processes from many domains, such as software and business processes. In the next section we present some basics regarding Petri nets and their subclass, the workflow nets.

4.2 Classical Petri nets. Workflow nets.

4.2.1 Petri nets

The Petri net was invented by Carl Adam Petri (Petri [1962]). We use a variant of the classic Petri net model, namely Place/Transition nets. Below we present some basics about classical Petri nets. For more details about classical Petri net notions, we refer to Desel and Esparza. [1995], Murata [1989], Reisig and Rosenberg [1998].

The classical Petri net is a directed graph with two node types called *places* and *transitions*. The nodes are connected via directed *arcs*. Connections between two nodes of the same type are not allowed. Places are represented by circles and transitions by rectangles (or by vertical bars).

Definition 1 (*P/T-nets*) A Place-Transition net (*P/T-net*) is a tuple (P, T, F) where:

- P is a finite set of places,
- T is a finite set of transitions ($P \cap T = \emptyset$),
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation).

A *marked P/T-net* is a pair (N, s) , where $(N = P, T, F)$ is a P/T-net and where s is a bag over P denoting the marking of the net. The set of all marked P/T-nets is denoted \mathcal{N} .

A marking is a bag over the set of places P , i.e. it is a function from P to the natural numbers. Square brackets for the enumeration of a bag are used, e.g., $[a^2, b, c^3]$ denotes the bag with two a -s, one b , and three c -s. The sum of two bags $(X + Y)$, the difference $(X - Y)$, the presence of an element in a bag ($a \in X$), and the notion of subbags ($X \leq Y$) are defined in a straightforward way and they can handle a mixture of sets and bags.

Let $N = (P, T, F)$ be a P/T-net. Elements $P \cup T$ are called *nodes*. A node x is an *input node* of another node y if and only if there is a directed arc from x to y (i.e. xFy). Node x is an *output node* of y if and only if yFx . For any $x \in P \cup T$, $\bullet^N x = \{y \mid yFx\}$ and $x \bullet^N = \{y \mid xFy\}$; the superscript N may be omitted if clear from the context. We use $\bullet t$ to denote the set of input places for a transition t . The notations $t \bullet$, $\bullet p$ and $p \bullet$ have similar meanings, e.g. $p \bullet$ is the set of transitions sharing p as an input place.

Figure 4.1 shows a P/T-net consisting of 10 places and 12 transitions. Transition a has one input place and one output place, transition f has one input place and two output places, transition k has two input places and one output place. The black dot in the input place of a represents a token, which denotes the initial marking. The dynamic behavior of such a marked P/T-net is defined by a *firing rule*.

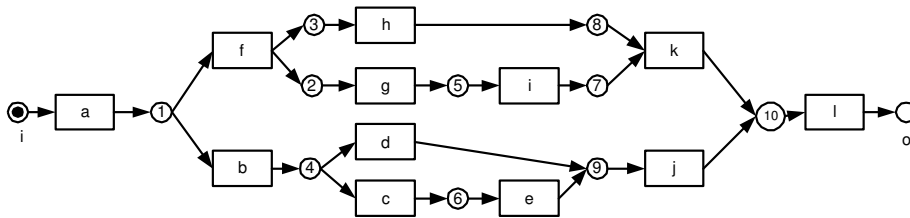


Figure 4.1: An example of Petri net

Definition 2 (Firing rule) Let $(N = (P, T, F), s)$ be a marked P/T-net. Transition $t \in T$ is *enabled*, denoted $(N, s)[t]$, if and only if $\bullet t \leq s$. The **firing rule** $[-]_- \subseteq \mathcal{N} \times T \times \mathcal{N}$ is the smallest relation satisfying for any $(N = (P, T, F), s) \in \mathcal{N}$ and any $t \in T$, $(N, s)[t] \Rightarrow (N, S) [t] (N, s - \bullet t + t \bullet)$.

In the marking shown in Figure 4.1 (i.e. one token in the source place), transition a is enabled and firing this transition removes the token from the input place and puts a token in the output place. In the resulting marking, the transitions b and f are enabled. Although both are enabled, only one can fire. If f fires, one token is consumed and two tokens are produced.

Definition 3 (Reachable markings) Let (N, s_0) be a marked P/T-net in \mathcal{N} . A marking s is *reachable* from the initial marking s_0 if and only if there exists a sequence of enabled transitions whose firing leads from s_0 to s . The set of reachable markings of (N, s_0) is denoted $[N, s_0]$.

The marked P/T-net shown in Figure 4.1 has 10 reachable markings. Sometimes it is convenient to know the sequence of transitions that are fired in order to reach some given markings. In this thesis we use the following notations for sequences. Let A be some alphabet of identifiers. A sequence of length n , for some natural number $n \in \mathbb{N}$, over alphabet A is a function $\sigma : \{0, \dots, n-1\} \rightarrow A$. The sequence of length zero is called the empty sequence and written ϵ . For the sake of readability, a sequence of positive length is usually written by juxtaposing the function values. For example, a sequence $\sigma = \{(0, a), (1, a), (2, b)\}$, for $a, b \in A$, is written aab . The set of all sequences of arbitrary alphabet length over alphabet A is written A^* .

Definition 4 (Firing sequence) Let (N, s_0) with $N = (P, T, F)$ be a marked P/T-net. A sequence $\sigma \in T^*$ is called a **firing sequence** of (N, s_0) if and only if, for some natural number $n \in \mathbb{N}$, there exist markings s_1, \dots, s_n and transitions $t_1, \dots, t_n \in T$ such that $\sigma = t_1, \dots, t_n$ and, for all i with $0 \leq i \leq n$, $(N, s_i)[t_{i+1}]$ and $s_{i+1} = s_i - \bullet t_{i+1} + t_{i+1} \bullet$.

Note that $n = 0$ implies that $\sigma = \epsilon$ and that ϵ is a firing sequence of (N, s_0) . Sequence σ is said to be enabled in marking s_0 , denoted $(N, s_0)[\sigma]$. Firing the sequence σ results into a marking s_n , denoted $(N, s_0)[\sigma] (N, s_n)$.

Definition 5 (Connectedness) A net $N = (P, T, F)$ is **weakly connected**, or simply **connected**, if and only if, for every two nodes x and y in $P \cup T$, $x(F \cup F^{-1})^*y$, where R^{-1} is the inverse and R^* the reflexive and transitive closure of a relation R . Net N is **strongly connected** if and only if, for every two nodes x and y , xF^*y .

We assume that all nets are weakly connected and have at least two nodes. The P/T-net shown in Figure 4.1 is connected but not strongly connected, because there is no directed path from sink place to the source place, or from l to a .

Definition 6 (Boundedness, safeness) A marked net $(N = (P, T, F), s)$ is **bounded** if and only if the set of reachable marking $[N, s]$ is finite. It is **safe** if and only if, for any $s' \in [N, s]$ and any $p \in P$, $s'(p) \leq 1$.

Note that safeness implies boundedness. The marked P/T-net shown in Figure 4.1 is safe (and therefore also bounded), because none of the 10 reachable states puts more than one token in a place.

Definition 7 (Dead transitions, liveness) Let $(N = (P, T, F), s)$ be a marked P/T-net. A transition $t \in T$ is **dead** in (N, s) if and only if there is no reachable marking $s' \in [N, s]$ such that $(N, s')[t]$. (N, s) is **live** if and only if, for every reachable marking $s' \in [N, s]$ and $t \in T$, there is a reachable marking $s'' \in [N, s']$ such that $(N, s'')[t]$.

Note that liveness implies the absence of dead transitions. None of the transitions in the marked P/T-net shown in Figure 4.1 is dead. However, the marked P/T-net is not live since it is not possible to enable each transition continuously.

4.2.2 Workflow nets

Petri nets (PN) have been used successfully to model and analyze processes from many domains, such as software and business processes, especially workflow processes. *Workflow processes* are case oriented, which means that each activity executed in the workflow corresponds to a case. Workflow management systems such as Staffware, IBM MQSeries, COSA, etc., have been developed to support structured business processes (van der Aalst et al. [2000], van der Aalst and van Hee. [2002]). These systems are provided with building blocks such as AND-split, AND-join, OR-split and OR-join in order to specify the routing of cases (Jablonski and Bussler [1996], van der Aalst [1998]). The routing in a workflow assumes four kinds of routing constructs: sequential, parallel, conditional and iterative routing (van der Aalst [1998]). Sequential routing concerns ordered causal relationships between tasks. For example, if we consider tasks A and B, we have a sequential routing construct when task B is executed only after task A is executed. Parallel routing is used when the order of execution is less strict.

Petri nets can be used to model the routing of cases: tasks are modelled by transitions and causal dependencies are modelled by places and arcs. A place corresponds to a condition which can be used as pre- and/or post-condition for tasks. An AND-split corresponds to a transition with two or more output places, and an AND-join corresponds to a transition with two or more input places. OR-splits/OR-joins correspond to places with multiple outgoing/ingoing arcs.

A parallel routing is modelled by AND-split and AND-join blocks. Conditional routing allows the modelling of a choice between two or more alternatives. To express the conditional construct, OR-split and OR-join blocks are used. In Figure 4.1 we can identify the following routing constructs: transitions f is an AND-split and k is an AND-join. Places 1 and 4 are OR-splits and places 9 and 10 are OR-joins.

A Petri net which models the control-flow dimension of a workflow process is called a WorkFlow net (WF-net).

Definition 8 (Workflow nets) Let $N = (P, T, F)$ be a P/T-net and \bar{t} a fresh identifier not in $P \cup T$. N is a **workflow net** (WF-net) if and only if:

1. *object creation*: P contains an input place i such that $\bullet i = \emptyset$,
2. *object completion*: P contains an output place o such that $o \bullet = \emptyset$,
3. *connectedness*: $\bar{N} = (P, T \cup \{\bar{t}\}, F \cup \{(o, \bar{t}), (\bar{t}, i)\})$ is strongly connected.

The P/T-net shown in Figure 4.1 is a WF-net. Note that although the net is not strongly connected, the short-circuited net with transition \bar{t} is strongly connected. Even if a net meets all the syntactical requirement stated in Definition 8, the corresponding process may exhibit errors such as deadlocks, tasks which can never become active, livelocks, garbage being left in the process after termination, etc. Therefore, we define the following correctness criterion.

Definition 9 (Sound) Let $N = (P, T, F)$ be a WF-net with input place i and output place o . N is **sound** if and only if:

1. *safeness*: $(N, [i])$ is safe,
2. *proper completion*: for any marking $s \in (N, [i])$, $o \in s$ implies $s = [o]$,
3. *option to complete*: for any marking $s \in (N, [i])$, $[o] \in [N, s)$, and
4. *absence of dead tasks*: $(N, [i])$ contains no dead transitions.

In other words: (i) the proper completion means that a case can be always completed, (ii) the option to complete means that after the completion of an activity, no work is left behind in the workflow and (iii) the absence of dead tasks requires the absence of states that cannot be reached. The set of all sound WF-nets is denoted \mathcal{W} .

The WF-net shown in Figure 4.1 is sound. Soundness can be verified using standard Petri-net-based analysis techniques. It was shown that a sound Petri net corresponds to the live and bounded corresponding short-circuited net (van der Aalst [1997, 1998]). Efficient algorithms for analyzing the WF-nets have been implemented in software tools as, for example, Woflan (Verbeek et al. [2001]).

Free-choice workflow nets are a special class of workflow nets. In van der Aalst [1998], the correspondence between the workflow management systems and the free-choice Petri nets are presented. Most of the workflow management systems abstract from states between tasks, i.e. states are not represented explicitly. Because of this, every choice is made inside an OR-split building block. An OR-split corresponds to a number of transitions sharing the same set of input places. This means that for these workflow management systems, a workflow procedure corresponds to a free-choice Petri net. Formally, we can express the free-choice property as follows:

Definition 10 (*Free choice*) A Petri net is a free-choice Petri net if and only if for every two transitions t_1 and t_2 , $\bullet t_1 \cap \bullet t_2 \neq \emptyset$ implies $t_1 = t_2$.

In other words, a Petri net is free-choice if and only if for every two transitions that share the same input place, the two corresponding input sets are the same. The free-choice Petri nets they have been studied extensively and strong theoretical results and efficient analysis techniques exist (van der Aalst [1998]).

4.3 The discovery problem

We introduced in Section 4.1 the basic idea of process discovery, i.e., as an alternative to hand-designing a process, we propose to collect the sequences of events produced over time by that process, and discover the underlying process model from these sequences. In this formal approach, we assume that it is possible to record events such that (i) each event refers to a task and (ii) each event refers to a case. We call a set of such recorded sequences the *process log*.

We make the assumption that it is possible to collect event data into process logs. These process logs are used to construct a process specification which adequately models the behavior registered. The term *process discovery* refers to methods for distilling a structured process description from a set of real executions. Because these methods focus on so-called case-driven process executions that are supported by contemporary workflow management systems, the term *workflow discovery* or *workflow mining* is also used (van der Aalst et al. [2002]).

To illustrate the idea of process discovery, consider the process log from Table 4.1. This log abstracts from the time, date, and event type, and limits the information to the order in which tasks are being executed. In this example, there are seven cases that have been processed; twelve different tasks occur in these cases. We can notice the following: for each case, the execution starts with task a and ends with task l , if c is executed, then e is executed. Also, sometimes we see task h and i after g and h before g .

Table 4.1: A process log example corresponding to the Petri net from Figure 4.1.

Case number	Executed tasks
Case 1	a f g h i k l
Case 2	a b c e j l
Case 3	a f h g i k l
Case 4	a f g i h k l
Case 5	a b c e j l
Case 6	a b d j l
Case 7	a b c e j l

Using the information shown in Table 4.1, we can discover the Petri net process model shown in Figure 4.1. In this simple example, the construction of the Petri net was straightforward. However, in the case of real-world processes where much more tasks are involved and with a high level of parallelism, the problem of discovering the underlying process becomes very complex. Moreover, the existence of noise into the log complicates the problem even more. The challenge of process mining is to derive “good” process models with as little information as possible.

Table 4.1 contains the *minimal information* we assume to be present. In many applications, the process log contains additional information as time stamps for each event, type of event (e.g., a start event, a complete event, a withdraw event) or information relating roles. In this thesis we will only consider the information recorded in process logs that refers to tasks that have been already executed. In van der Aalst and van Dongen [2002] the discovery algorithm is extended to incorporate timing information.

Definition 11 (Process trace, Process log) Let T be a set of tasks. $\delta \in T^*$ is a process trace and $W \in \mathcal{P}(T^*)$ is a process log³.

An example of a process log is given in Table 4.1. A process trace for case 1 is “ a, f, g, h, i, k, l ”. Inspecting the process log presented in Table 4.1 we can notice that the traces for cases 1, 3, 4 and 6 appear in the log just once, while the trace “ a, b, c, e, j, l ” appears three times, e.g. for cases 2, 5 and 7. The event frequencies are extremely important when the log contains noisy data. In this chapter we consider that noise-free data are recorded in the process log, while in Chapter 5 we assume that there is noise in the process log.

³ $\mathcal{P}(T^*)$ is the powerset of T^* , i.e. $W \subseteq T^*$. T^* is the set of all sequences that are composed of zero or more tasks of T .

We introduce the following notions in order to simplify the use of log and sequences ⁴:

Definition 12 (*∈, first, last*): Let A be a set, $a \in A$, and $\sigma = a_1a_2\dots a_n \in A^*$ a sequence over A of length n . $\in, first, last$ are defined as follows:

1. $a \in \sigma$ if and only if $a \in \{(a_1, a_2, \dots, a_n)\}$
2. $first(\sigma) = a_1$, and
3. $last(\sigma) = a_n$.

Our method of discovering the Petri net process model from a log file is based on finding the relations that can exist between tasks. For example, if a task is always followed by another task, it is likely that there is a causal relation between both tasks. We define the log-based relations as follows:

Definition 13 (*Log-based ordering relations*) Let W be a process log over T , i.e. $W \in \mathcal{P}(T^*)$ and $x, y \in T$. The following relations are defined:

- the **succession relation**, $x >_W y$: if and only if there is a trace $\sigma = t_1t_2\dots t_n$ and $i \in \{1, \dots, n-1\}$ such that $\sigma \in W$ and $t_i = x$ and $t_{i+1} = y$,
- the **causal relation**⁵, $x \rightarrow_W y$: if and only if $x >_W y$ and $y \not>_W x$,
- the **exclusive relation**, $x \#_W y$: if and only if $x \not>_W y$ and $y \not>_W x$,
- the **parallel relation**, $x \parallel_W y$: if $x >_W y$ and $y >_W x$.

To illustrate the above definitions, let us consider again the process log from Table 4.1 corresponding to the Petri net from Figure 4.1. The succession relation $>_W$ describes which tasks appeared in sequence, i.e. one directly following the other. In the log from Table 4.1, $a >_W f$, $f >_W g$, $b >_W c$, $h >_W g$, $g >_W h$, etc. There are three possible situations in which a pair of events can be:

1. events c and e are in sequence: then $c >_W e$, $e \not>_W c$, thus $c \rightarrow_W e$;
2. there is a choice between events b and f : then $b \not>_W f$, $f \not>_W b$, thus $b \#_W f$ (and $f \#_W b$);
3. events h and i are in parallel: then $h >_W i$, $i >_W h$, thus $h \parallel_W i$ (and $i \parallel_W h$).

⁴We use the formalizations for these notions and for the log-based relations from van der Aalst et al. [2003, to appear].

⁵When using the term “causal” relation, we are aware of the problem that via induction from data it is difficult to distinguish causal relationship from correlation, and there might be an unknown latent variable that is the real cause. Consider the situation where making a product requires two tasks x and y to be completed. These tasks are independent from each other, but the company *chooses* to complete them always in the same order, e.g. x before y . In our view, a causal relation *exists* between tasks x and y , not necessarily because x is causing y , rather because there is a direct succession between x and y (x is always completed right before y).

In van der Aalst et al. [2002] was shown that relations \rightarrow_W , \rightarrow_W^{-1} , $\#_W$ and \parallel_W are mutually exclusive and partition $T \times T$ ⁶.

Given these initial definitions, in the next section we present an algorithm for discovering the Petri net process model. An analysis of the classes of process nets that can be rediscovered with this algorithm from a process log is described in Section 4.3.2.

4.3.1 The algorithm for discovering process models

In this section we describe the algorithm for discovering the Petri net process model from a process log⁷.

An essential notion in relation to the quality of the discovered process is the *completeness* of a log. If the process is complex, a small set of traces will be not enough to discover the exact behavior of the process. It is unrealistic to consider that all possible firing sequences are present in a process log. One reason is that in case of loops, the number of possible sequences may be infinite. Another reason is that parallel events typically have an exponential number of states and subsequently, so the number of possible firing sequences may become very large.

Because our algorithm is based on the relations \rightarrow_W , \rightarrow_W^{-1} , $\#_W$ and \parallel_W and since they can be derived from the $>_W$ relation, we assume the log to be complete with respect to $>_W$ relation⁸.

Definition 14 (Complete process log) Let $N = (P, T, F)$ be a sound WF-net, i.e., $N \in \mathcal{N}$. W is a process log of N if and only if $W \in \mathcal{P}(T^*)$ and every trace $\sigma \in W$ is a firing sequence of N starting in state $[i]$, i.e., $(N, [i])[\sigma]$. W is a **complete** process log of N if and only if (i) for any process log W' of N : $>_{W'} \subseteq >_W$, and (ii) for any $t \in T$ there is a $\sigma \in W$ such that $t \in \sigma$.

A process log of a sound Petri net only contains behaviors that can be exhibited by the corresponding process. A process log is complete if all tasks that potentially directly follow each other in fact directly follow each other in some trace from the log.

Definition 15 (The α process discovery algorithm) Let W be a process log over T . $\alpha(W)$ is defined as follows:

1. $T_W = \{t \in T \mid \exists \sigma \in W t \in \sigma\}$,
2. $T_I = \{t \in T \mid \exists \sigma \in W t = \text{first}(\sigma)\}$,
3. $T_O = \{t \in T \mid \exists \sigma \in W t = \text{last}(\sigma)\}$,
4. $X_W = \{(A, B) \mid A \subseteq T_W \wedge B \subseteq T_W \wedge \forall a \in A \forall b \in B a \rightarrow_W b \wedge \forall a_1, a_2 \in A a_1 \#_W a_2 \wedge \forall b_1, b_2 \in B b_1 \#_W b_2\}$,
5. $Y_W = \{(A, B) \mid \exists (A', B') \in X_W A \subseteq A' \wedge B \subseteq B' \implies (A, B) = (A', B')\}$,

⁶We consider the inverse of the causal relation \rightarrow_W^{-1} , i.e. $\rightarrow_W^{-1} = \{(y, x) \in T \times T \mid x \rightarrow_W y\}$.

⁷We are using the same definitions and the formal description of the discovery algorithm as presented in van der Aalst et al. [2002].

⁸For an in-depth discussion about the notion of completeness with respect to succession relation $>_W$, see van der Aalst et al. [2003, to appear].

6. $P_W = \{p_{(A,B)} \mid (A, B) \in Y_W\} \cup \{i_W, o_W\}$,
7. $F_W = \{a, p_{(A,B)} \mid (A, B) \in Y_W \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_W \wedge b \in B\} \cup \{(i_W, t) \mid t \in T_I\} \cup \{(t, o_W) \mid t \in T_O\}$, and
8. $\alpha(W) = (P_W, T_W, F_W)$.

The discovery algorithm constructs a net (P_W, T_W, F_W) . The set of transitions T_W can be derived by inspecting the log. Since it is possible to find all initial transitions T_I and all final transitions T_O , it is easy to construct the connections between these transitions and i_W and o_W . Besides the source place i_W and the sink place o_W , places of the form $p_{(A,B)}$ are added. For such place, the subscript refers to the set of input and output transitions, i.e., $\bullet p_{(A,B)} = A$ and $p_{(A,B)} \bullet = B$. A place is added in-between a and b if and only if $a \rightarrow_W b$. However, in case of OR-splits/joins, some of these places need to be merged. For this purpose, the relations X_W and Y_W are constructed. $(A, B) \in X_W$ if there is a causal relation from each member of A to each member of B and the members of A and B never occur next to each other. Note that if $a \rightarrow_W b$, $b \rightarrow_W a$, or $a \parallel_W b$, then a and b cannot be both in A (or B). Relation Y_W is derived from X_W by taking only the largest elements with respect to set inclusion.

To illustrate our algorithm, we consider the process net shown in Figure 4.1 that generated the log presented in Table 4.1. Note that this log is complete, according to the notion of completeness specified in Definition 14. Using this log, we want to discover the underlying process net. The basic idea of our approach is to connect (i) all events $x \rightarrow_W y$ and to add a place between x and y and (ii) to merge those places whether there is an OR-split/join, and leave the places unchanged when there is an AND-split/join. For the second step, we use the relations $x \#_W y$ and $x \parallel_W y$. From the log presented in Table 4.1, we can infer the following causal relations, by applying Definition 13: $a \rightarrow_W f$, $a \rightarrow_W b$, $f \rightarrow_W g$, $f \rightarrow_W h$, $b \rightarrow_W d$, $b \rightarrow_W c$, $g \rightarrow_W i$, $h \rightarrow_W k$, $i \rightarrow_W k$, $c \rightarrow_W e$, $e \rightarrow_W j$, $d \rightarrow_W j$, $k \rightarrow_W l$, $j \rightarrow_W l$. According to our algorithm, we add on each arc one place (represented as a small circle), as shown in Figure 4.2. The next step is to merge those places in case of OR-splits/joins, by using the relations $x \#_W y$. Thus, we have to perform four merge tasks, i.e. to merge:

1. the two places from event a to f and from a to b , ($b \# f$),
2. the two places from event b to d and from b to c , ($c \# d$),
3. the two places from event d to j and from e to j , ($d \# e$),
4. the two places from event k to l and from j to l , ($k \# j$).

In Figure 4.2, the places that need to be merged are marked by dotted ellipses. After merging, we recover the Petri net from Figure 4.1.

In Mărușter et al. [2001, 2002a] we evaluated the initial version of the discovery algorithm by testing five different sound and acyclic Petri nets. The Petri nets were similar with the example given in Figure 4.1, i.e. they contain between 10-12 transitions, involving parallel, conditional and sequence routing constructs. For each Petri net, we generated random logs with 500 event traces. In four experiments, the resulting Petri nets have the same structure as the original Petri nets. However, in

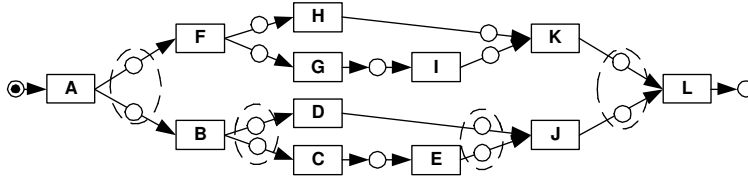


Figure 4.2: The directed graph corresponding to the log presented in Table 4.1, that contains the events in relation $x \rightarrow_W y$, after adding places. The places that need to be merged, according to the α algorithm, are marked by dotted ellipses.

the fifth experiment involving a not-free-choice Petri net, some causal relations are missed.

Given these results, the following question arises: what is the class of relevant Petri net that can be rediscovered using the α algorithm and in which conditions? In the following section the answer to this question is provided.

4.3.2 Which processes can be rediscovered?

In this section we present the rediscovery problem, formulated as “what is the class of sound process nets that α algorithm can rediscover, on the basis of complete process logs”?⁹

Suppose that we have a log based on many executions of the process described by a Petri net PN_1 . Based on this process log and using a discovery algorithm, we construct a Petri net PN_2 . The question is whether $PN_1 = PN_2$. In this section, the class of Petri nets for which $PN_1 = PN_2$ is explored.

We showed in the previous section that, according to the α algorithm, a place is added to connect two transitions, whenever a causal relation existed between these two transitions. In van der Aalst et al. [2002], the relation between the causal relations detected in the log (i.e., \rightarrow_W) and the presence of places connecting transitions has been inspected. In this paper it has been proven that under the assumptions of a sound process net and a complete process log, causal relations imply connecting places (Theorem 4.1 in van der Aalst et al. [2002]). Second, the class has been identified for which connecting places imply causal relations. For this class, certain requirements have been specified, as we present in Definition 16.

Definition 16 (SWF-nets) A Petri net $N = (P, T, F)$ is an SWF-net (Structured Workflow net) if and only if:

1. for all $p \in P$ and $t \in T$, with $(p, t) \in F : |p \bullet| > 1$ implies $|\bullet t| = 1$.
2. for all $p \in P$ and $t \in T$, with $(p, t) \in F : |\bullet t| > 1$ implies $|\bullet p| = 1$.
3. There are not implicit places.

⁹The analysis relating the class of process models for which it is possible to accurately rediscover the model is reproduced from van der Aalst et al. [2003, to appear].

The Petri net presented in Figure 4.1 contain no implicit places. However, adding a new place p connecting transition c and e yields an implicit place. It is not possible to construct a discovery algorithm that is able to detect the place p , since the addition of the place does not change the behavior of the net and therefore is not visible in the log. For the rediscovery problem it is very important that the structure of the Petri net clearly reflects its behavior. Therefore, we eliminate constructs as shown in Figure 4.3. The left construct illustrates the constraint that choice and synchronization should never meet. If two transitions share an input place, and thus “fight” for the same token, they should not require synchronization. This means that choices (places with multiple output transitions) should not be mixed with synchronization. The right-hand construct from Figure 4.3 illustrates the constraint that if there is synchronization, all preceding transitions should have fired, i.e. it is not allowed to have synchronization directly preceded by an OR-join. Petri nets that satisfy these requirements are called *structured workflow nets*.

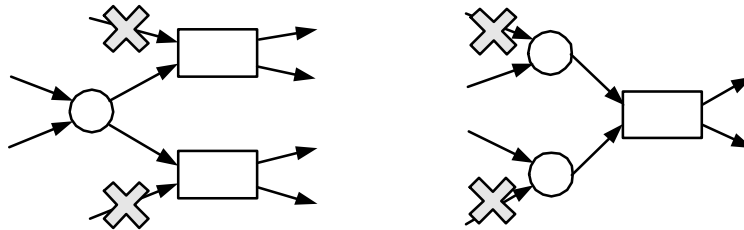


Figure 4.3: Two constructs not allowed in SWF-nets.

Looking at the three requirements specified in Definition 16, they seem to be quite restrictive. From a practical point of view, this is not the case. First of all, SWF-nets allow for all routing constructs encountered in practice, i.e. sequential, parallel, conditional and iterative routing are possible and the basic process building blocks (AND-split, AND-join, OR-split and OR-join) are supported. Second, Petri nets that are not SWF-nets are difficult to understand and they are hardly interpretable in the context of process management systems, where only processes that corresponds to SWF-nets are allowed. The latter observation can be explained by the fact that most process management systems use a language with separate building blocks for OR-splits and AND-joins. Finally, there is a very pragmatic argument. If we drop any of the requirements stated in Definition 16, relation $>_W$ does not contain enough information to successfully mine all processes in the resulting class.

The reader familiar with Petri nets will notice that SWF-nets belong to the class of free-choice Petri nets (Desel and Esparza. [1995]). This allow us to use efficient analysis techniques and advanced theoretical results, e.g. we can decide soundness in polynomial time (van der Aalst [1998]).

The property that follows from Definition 16 is that two transitions cannot be connected by multiple places. Formally, we describe this as follows:

Property 1 *Let $N = (P, T, N)$ be an SWF-net. For any $a, b \in T$ and $p_1, p_2 \in P$: if $p_1 \in a \bullet \cap \bullet b$ and $p_2 \in a \bullet \cap \bullet b$, then $p_1 = p_2$.*

This property illustrates that the structure of an SWF-net clearly reflects its behavior and vice versa.

We already mentioned that in van der Aalst et al. [2002] it has been proven that causal relations imply the presence of places. In order to show the reverse idea for SWF-nets, (i.e. that the presence of places imply causal relation), first it has been proven in van der Aalst et al. [2002] that the presence of places implies the existence of the $>_W$ relation. However, the presence of places does not imply the existence of the \rightarrow_W relation. To illustrate this, consider Figure 4.4. For the first two nets (i.e. N_1 and N_2), two tasks are connected if and only if there is a causal relation. This does not hold for N_3 and N_4 . In N_3 , $A \rightarrow_{W_3} B$, $A \rightarrow_{W_3} D$, $B \rightarrow_{W_3} D$. However, not $B \rightarrow_{W_3} B$. Nevertheless, there is a place connecting B to B . In N_4 , although there are places connecting B to C and vice versa, $B \not\rightarrow_{W_4} C$ and $C \not\rightarrow_{W_4} B$. These examples indicate that loops of length one (see N_3) and length two (see N_4) are harmful. However, loops of length three or longer are no problem (see also Theorem 4.6 from van der Aalst et al. [2002]).

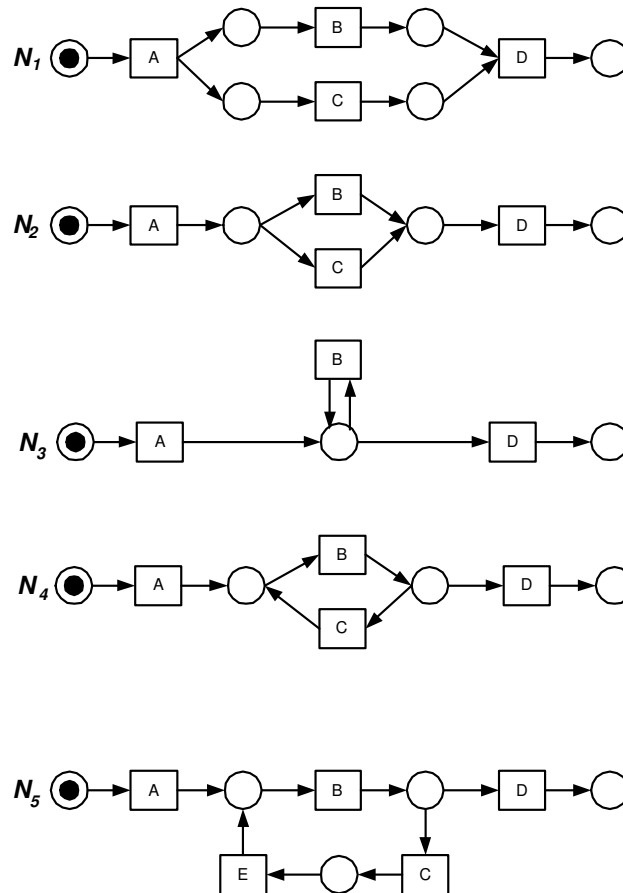


Figure 4.4: Five sound SWF-nets.

Let us now turn to the rediscovery problem. Is it possible to rediscover Petri nets using α algorithm? Consider the five nets shown in Figure 4.4. If α algorithm is applied to a complete process log of N_1 , the resulting net is N_1 modulo renaming of places. Similarly, if α algorithm is applied to a complete process log of N_2 , the resulting net is N_2 modulo renaming of places. As expected, the α algorithm is not able to rediscover N_3 and N_4 . $\alpha(W_3)$ is not a Petri net since B is not connected to the rest of the net. $\alpha(W_4)$ is not a Petri net since C is not connected to the rest of the net. In both cases, two arcs are missing in the resulting net. N_3 and N_4 illustrate that the mining algorithm is unable to deal with short loops. Loops of length three or longer are no problem. For example, $\alpha(W_5) = N_5$ modulo renaming of places. It can be proven that the α algorithm is able to rediscover the class of SWF-nets given that there are no short loops (see van der Aalst et al. [2002]). Formally, this main result is presented in Theorem 4.3.1.

Theorem 4.3.1 *Let $N = (P, T, F)$ be a sound SWF-net and let W be a complete process log of N . If for all $a, b \in T$, $a \bullet \cap \bullet b = \emptyset$ or $b \bullet \cap \bullet a = \emptyset$, then $\alpha(W) = N$ modulo renaming of places.*

Nets N_1 , N_2 and N_5 shown in Figure 4.4 satisfy the requirements stated in Theorem 4.3.1. Therefore, it is not surprising that the α algorithm is able to rediscover these nets. The net shown in Figure 4.5a is also an SWF-net with no short loops. Therefore, we can successfully rediscover the net if the AND-split and the AND-join are *visible in the log*. The latter assumption is not realistic if these two transitions do not correspond to real work. Given the log $W = \{ABCD, ACBD, AED\}$, we cannot see the occurrences of AND-split and AND-join, thus we can consider them invisible. However, if we apply α to the log W , then the result is quite surprising. The resulting net $\alpha(W)$ is shown in Figure 4.5 b. Although the net is not an SWF-net, it is a sound Petri net whose observable behavior is identical to the net shown in Figure 4.5a. Also note that the Petri net shown in Figure 4.5b can be rediscovered, although it is not an SWF-net. This example shows that the applicability is not limited to SWF-nets. However, for arbitrary sound Petri nets it is not possible to guarantee that they can be rediscovered.

To conclude this analysis, we revisit the first two requirements in Definition 16. We referred earlier to the motivation of restricting ourselves to SWF-nets. To illustrate the necessity of these requirements, consider Figures 4.6 and 4.7. The Petri net N_6 shown in Figure 4.6 is sound, but it is not an SWF-net, since the first requirement is violated (N_6 is not free-choice). If we apply the mining algorithm to a complete process log W_6 of N_6 , we obtain the Petri nets N_7 also shown in Figure 4.6 (i.e. $\alpha(W_6) = N_7$). Clearly, N_6 cannot be rediscovered using α . Although N_7 is a sound SWF-net, its behavior is different from N_6 , e.g. the process trace ACE is possible in N_7 , but not in N_6 . This example motivates the first requirement in Definition 16. The second requirement is motivated by Figure 4.7. N_8 violates the second requirement. If we apply the mining algorithm to a complete process log W_8 of N_8 , we obtain the Petri net $\alpha(W_8) = N_9$, also shown in Figure 4.7. Although N_9 is behaviorally equivalent, N_8 cannot be rediscovered using the discovery algorithm.

Although the requirements stated in Definition 16 are necessary in order to prove that this class of process processes can be rediscovered on the basis of a complete

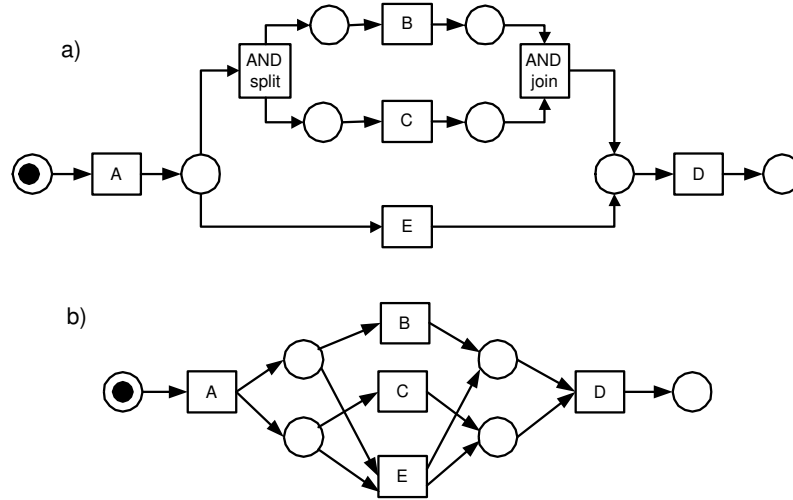


Figure 4.5: A process net with invisible AND-split and AND-join a), and the corresponding discovered workflow net b).

process log, the applicability is not limited to SWF-nets. The examples given in this section show that in many situations a behaviorally equivalent Petri net can be derived. Even in the cases where the resulting Petri net is not behaviorally equivalent, the results are meaningful, e.g. the process represented by N_7 is different from the process represented by N_6 (cf. Figure 4.6). Nevertheless, N_7 is similar and captures most of the behavior.

In van der Aalst et al. [2003, to appear], the limitation of the α algorithm in dealing with short loops is discussed in depth and some possible solutions are indicated, e.g. to use a stronger notion of completeness.

4.4 Process discovery literature

The idea of process discovery is not new (Agrawal et al. [1998], Cook and Wolf [1998a,b], Cook and Wolf. [1999], Herbst [2000a,c, 2001], Herbst and Karagiannis [1998, 1999, 2000], Maxeiner et al. [2001], Schimm [2000a,b, 2001a,b, 2002], Weijters and Aalst [2001a,b, 2002]). Cook and Wolf have investigated similar issues in the context of software engineering processes. In Cook and Wolf [1998a], process discovery is viewed as a problem of grammar inference, similar with the discovery of a grammar for a regular language (Angluin and Smith [1983]). Cook and Wolf describe three methods for process discovery that result into finite-state machine models: one using neural networks, one using a “purely algorithmic approach” (i.e. based solely on the finite-state machine approach), and one Markovian approach, that builds a finite state machine model using probabilities of sequences of events. The authors consider the latter two the most promising approaches. The purely algorithmic approach builds a finite-state machine model where states are fused if their futures (in terms of possible behavior in the next k steps) are identical. The Markovian approach uses a mixture

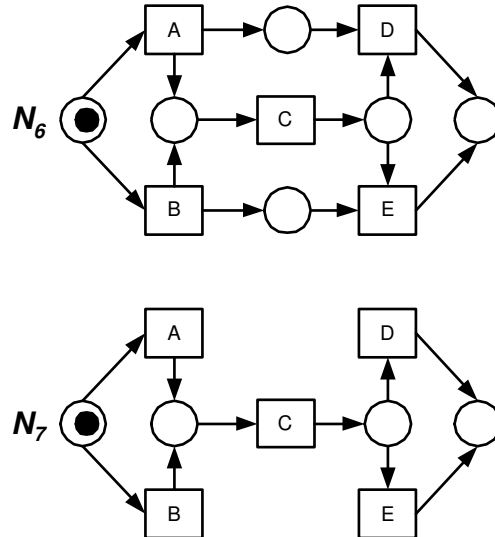


Figure 4.6: The non-free-choice Petri net N_6 cannot be rediscovered.

of algorithmic and statistical methods and is able to deal with noise. However, the results presented in Cook and Wolf [1998a] are limited to sequential behavior. That is an important practical disadvantage, because in many real business processes, activities occur in parallel. Cook and Wolf extend their work to concurrent processes (Cook and Wolf [1998b]). They propose specific metrics (entropy, event type counts, periodicity, and causality) and use these metrics to discover models out of event streams. However, they do not provide an approach to generate explicit process models. In Cook and Wolf. [1999], the authors provide a measure to quantify discrepancies between a process model and the actual behavior as registered using event-based data. The idea of applying process discovery in the context of workflow management was first introduced in Agrawal et al. [1998]. This work is based on *workflow graphs*, which are inspired by workflow products such as IBM MQSeries workflow (formerly known as Flowmark) and InConcert. A workflow graph is used to model a workflow, and it is represented as a directed graph, containing two types of objects: *nodes* and *control flow*. The nodes represent the work to be done in the workflow, and the control flow links two nodes in the graph. In Maxeiner et al. [2001], a tool based on the previous mentioned approach is presented. Schimm (Schimm [2000a,b, 2002]) has developed a mining tool suitable for discovering hierarchically structured workflow processes. This requires all splits and joins to be balanced. Herbst and Karagiannis also address the issue of process discovery in the context of workflow management (Herbst [2000a,c, 2001], Herbst and Karagiannis [1998, 1999, 2000]) using an inductive approach. The work presented in Herbst and Karagiannis [1998, 2000] is limited to sequential models. The approach described in Herbst [2000a,c, 2001], Herbst and Karagiannis [1999] also allows for concurrency. It uses stochastic task graphs as an intermediate representation and it generates a workflow model described in the ADONIS modelling language (Junginger et al. [2000]). In the induction step, task nodes are merged and split in order to discover the underlying process. An important difference with other

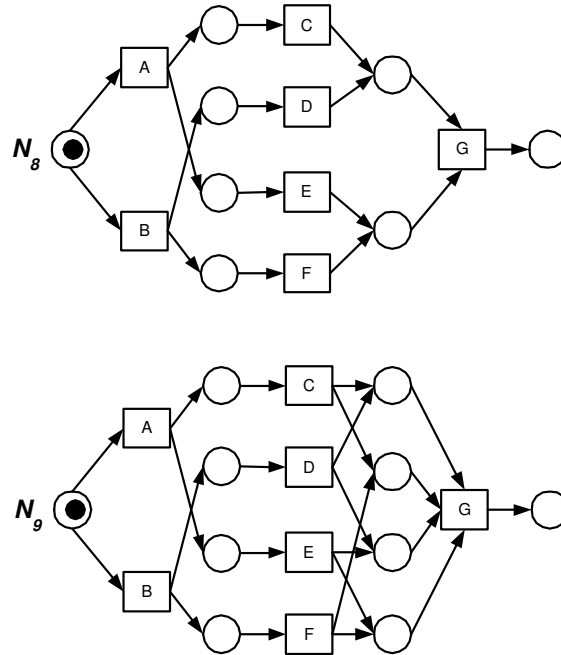


Figure 4.7: Petri net N_8 cannot be rediscovered. Nevertheless α return a Petri net which is behaviorally equivalent.

approaches is that the same task can appear multiple times in the workflow model. The graph generation technique is similar to the approach of Agrawal et al. [1998], Maxeiner et al. [2001]. The nature of splits and joins (i.e., AND or OR) is discovered in the transformation step, where the stochastic task graph is transformed into an ADONIS workflow model with block-structured splits and joins.

Process mining/discovery can be seen as a tool in the context of Business (Process) Intelligence (BPI). In Grigori et al. [2001], Sayal et al. [2002], a BPI tool-set that includes a so-called “BPI Process Mining Engine” is presented. This tool uses generic mining tools such as SAS Enterprise Miner for the generation of decision trees relating attributes of cases. For mining a process, it is convenient to have a so-called “process data warehouse” to store audit trails. A data warehouse simplifies and speeds up the queries needed to derive causal relations. In Eder et al. [2002], Mühlen [2001a,b], Mühlen and Rosemann. [2000], the design of such warehouse and related issues are discussed in the context of process logs. Also, Mühlen and Rosemann. [2000] describes the PISA tool which can be used to extract performance metrics from process logs. Similar diagnostics are provided by the ARIS Process Performance Manager (PPM) (Scheer [2002]). The later tool is commercially available and a customized version of PPM is the Staffware Process Monitor (SPM) (Staffware [2002]) which is tailored towards mining Staffware logs. Note that none of the latter tools is extracting the process model. The main focus is on clustering and performance analysis rather than causal relations as in Agrawal et al. [1998], Cook and Wolf [1998a,b], Cook and Wolf. [1999], Herbst [2000a,c, 2001], Herbst and Karagiannis [1998, 1999, 2000], Maxeiner

et al. [2001], Schimm [2000a,b, 2001a,b, 2002], van der Aalst et al. [2002].

Our approach on process discovery can be distinguished from the approaches previously mentioned in some aspects. Our focus is on discovering processes with concurrent behavior, rather than concentrating on finding ad-hoc mechanisms to capture parallelism. The approach described in this chapter differs from the already mentioned literature in the sense that for the α algorithm it is proven that for certain subclasses it is possible to find the right process model. In van der Aalst and van Dongen [2002], the α algorithm is extended to incorporate timing information. A comprehensive survey relating the newest issues and approaches in workflow mining is presented in van der Aalst et al. [2003].

However, in the literature previously mentioned, the issue of discovering a process model when data is noisy was not explicitly tackled.

4.5 Conclusions

In this chapter we answered to the fourth research question specified in Section 1.2 from Chapter 1, i.e.:

4. What kind of processes can be discovered from past process executions?

We provided a discovery algorithm that constructs a Petri net model from a noise-free process log that contains sufficient information (i.e., all tasks that potentially directly follow each other in fact directly follow each other in some trace from the log). Performing some initial experiments, the discovered models are Petri nets that have the same structure as the original Petri net models. However, in the experiment involving a non free-choice Petri net, some causal relations are missed. These results arouse the interest to investigate the limits of the discovery algorithm.

The rediscovery problem investigates whether using the α algorithm, it is possible to rediscover the process model, i.e., for which class of process models it is possible to accurately construct the model by looking at their logs. It has been shown that it is impossible to rediscover the class of all Petri nets, but the α algorithm can successfully rediscover a large class of relevant Petri nets, under the circumstances of a noise-free and complete log. An interesting characteristic of the α algorithm is that it constructs the “simplest” Petri net generating the behavior exhibited in the log.

The limitations of the α algorithm are the following. First, it cannot deal with non-free-choice constructs. It is also known in the Petri net literature that a lot of undecidable problems for general Petri nets are decidable in case of free-choice Petri nets. The second limitation of this algorithm is that short-loops cannot be depicted considering the current version of the completeness notion. Ideas to overcome the short-loop problem focus on considering a stronger notion of completeness.

The analysis given in this chapter have indicated the strengths, but also the boundaries of the discovery algorithm. Although a large class of process models can be successfully discovered, the main limitation of the α algorithm is that it cannot provide a process model in case of noisy and incomplete data. Unfortunately, this tends to be the case when dealing with real data. In the next chapter, we provide solutions for discovering the process model assuming that process log data are noisy and incomplete.

Chapter 5

The practical approach

In the previous chapter we saw how sequence data can be used in process modelling, assuming that there is no noise in the data and there is sufficient information.

However, in practical situations it seems realistic to assume that process logs contain noise. Noise can have different causes, such as missing registration data or input errors. Moreover, the log can contain insufficient information. In such situations, the discovery problem becomes more problematic. The proposed alternative is to employ statistical and machine learning techniques to induce predictive models that can be used to construct the Petri net process model¹.

We aim to respond to the fifth research question mentioned in Section 1.2 from Chapter 1, i.e.:

5. It is possible to extract process models from business data?

The structure of this chapter is as follows: in Section 5.1 we discuss about other process discovery work existing in the literature that deals with noisy process logs. We briefly introduce the problem statement of this chapter and the proposed solution in Section 5.2. The methodology of generating experimental data that serves to induce predictive models is presented in Section 5.3. In Section 5.4 the log relations that can exist between two tasks and the metrics for predicting these relations are presented. Two types of predictive models and their evaluation are described in Section 5.5. In Section 5.6 we discuss the influence of process characteristics on the models' performance. We conclude this chapter with some conclusions in Section 5.7.

5.1 Introduction

We provided in Section 4.4 an overview of the work done in the area of process discovery from process logs. Although substantial efforts have been invested towards process discovery, few research about discovering a process model from noisy logs could be found.

¹The content of this chapter has been appeared in Mărușter et al. [2002b] and also has been submitted (Mărușter et al. [2003]).

Cook and Wolf [1998a] describe three methods for process discovery that result into finite-state machine models: one using neural networks, one using a “purely algorithmic approach”, where states are fused if their futures (in terms of possible behavior in the next k steps) are identical, and one Markovian approach. The Markovian approach build a finite state machine model using probabilities of sequences of events and is able to deal with noise.

In Weijters and Aalst [2001a,b], an approach using rather simple metrics is used to construct the so-called “dependency/frequency tables”. Employing the information from the “dependency/frequency tables” and using some hand crafted threshold values, a “dependency/frequency graphs” is built, that finally results into a Petri net model. However, in some situations, the method is not robust enough for discovering the correct process model. In this chapter we find other metrics in addition to that described in Weijters and Aalst [2001a,b], we combine all these metrics and we induce models from logs for building the Petri net process model, as described in the next section.

5.2 Problem statement

As said, in practical situations it seems realistic to assume that process logs contain noise. Noise can have different causes, such as missing registration data or input errors. Moreover, the log can contain insufficient information. Another source of problems is the existence of imbalances between the task execution priorities, i.e. some tasks are more likely to be executed than others.

We aim to discover the Petri net process model from a noisy process log by finding the relations that can exist between tasks. For example, if a task is always followed by another task, it is likely that there is a causal relation between both tasks. We concentrate on finding the log-based relations already introduced in Section 4.3 (i.e. the causal relation \rightarrow_W , the exclusive relation $\#_W$ and the parallel relation \parallel_W).

Our approach is summarized below:

1. Find the log-based relations
 - (a) Develop relational metrics
 - (b) Induce models to predict the log-based relations, using the relational metrics. We concentrate on two type of models:
 - a logistic regression model
 - a rule-based model
2. Knowing the log-based relations, use the α process discovery algorithm (see Section 4.3.1) to construct the Petri net process model.

In order to induce robust models for predicting the log-based relations, we must have learning material. We can (i) collect real data or (ii) simulate data by varying process and log characteristics. The advantage of using simulating data is that we can manipulate these characteristics to see how they affect the output measures of performance. Therefore, we will generate an experimental material where different characteristics of the process logs are varied. In the rest of this chapter we describe

the approach proposed above. In Chapter 6 we will also test this approach on data resulting from real settings.

5.3 Experimental setting and data generation

The learning material that we use to induce models for predicting the log-based relations should resemble realistic process logs. From the possible log and process characteristics that vary from process to process and subsequently affect the process log, we identified four: (i) the total number of task types, (ii) the amount of available information in the process log, (iii) the amount of noise and (iv) the execution priorities in OR-splits and AND-splits.

Our experimental setting consists of variations of four process log characteristics:

The number of task types: we generate Petri nets with 12, 22, 32 and 42 task types.

The amount of information in the process log or log size: the amount of information is expressed by varying the number of lines (one line or trace represents the processing of one case). We consider logs with 200, 400, 600, 800 and 1000 lines.

The amount of noise: we generate noise performing four different operations, (i) delete the head of an event sequence, (ii) delete the tail of a sequence, (iii) delete a part of the body and (iv) interchange two randomly chosen tasks. During the noise generation process, minimally one event and maximally one third of the sequence is deleted. We generate five levels of noise: 0% noise (the common workflow log), 5% noise, 10%, 20% and 50% (we select 5%, 10%, 20% and respectively 50% of the original event sequences and we apply one of the four above described noise generation operations).

The imbalance of execution priorities: we assume that tasks can be executed with priorities between 0 and 2. In Figure 5.1, after executing task *a* (which is an OR-split), it is possible to exist an imbalance between executing task *b* and task *f*. For example, task *b* can have an execution priority of 0.8 and task *f* 1.5. This implies that after *a*, in 35 percent of the cases task *b* is selected, and in 65 percent of the cases, task *f* is executed.

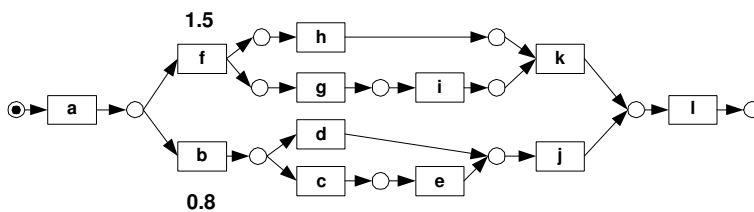


Figure 5.1: A Petri net process model example

The execution imbalance is produced on four levels:

- level 0, no imbalance: all tasks have the execution priority 1;

- level 1, small imbalance: each task can be executed with a priority randomly chosen between 0.9 and 1.1;
- level 2, medium imbalance: each task can be executed with a priority randomly chosen between 0.5 and 1.5;
- level 3, high imbalance: each task can be executed with a priority randomly chosen between 0.1 and 1.9.

First, we design four types of Petri nets: with 12, 22, 32 and 42 task types. Second, for each type of Petri net, we produce four unbalanced Petri nets, corresponding to the four levels of execution imbalance. Third, for each resulting Petri net, we generate a log file with 0%, 5%, 10%, 20% and 50% noise. Fourth, we vary the amount of information, i.e. we vary the number of lines in the log: each resulting noisy log is partitioned, considering the first 20% lines, then the first 40%, and so on, until 100% of material is considered. In the next section we show how the generated material is used to detect the log-based relations.

The idea of having “sufficient” information available in the log to extract a correct process model, can be expressed using the notions of log size, completeness and imbalance, which are all interconnected.

In Section 4.3.1 we expressed the concept of “sufficient” data by formally defining the notion of completeness with respect to $>_W$ relation. E.g., a process log is *complete* if all tasks that potentially follow directly each other, in fact they will directly follow each other in some trace in the log. However, in the theoretical approach, we did not consider at all the task frequencies. Also, in the theoretical approach, the log size was not an issue, but in the practical approach, the task frequencies are the basis of this approach.

For the practical approach, having not “sufficient” data could be the result of:

1. the log is incomplete with respect to $>_W$ relation. Even if the log contains a lot of information (i.e. many lines in the log) and the priorities of task executions are perfectly balanced, we cannot discover a correct model (because the log is incomplete). This was already discussed in Chapter 4.
2. the priorities of task executions are unbalanced. Actually, this affects the log completeness. If we assume the log to be complete and if the log size is big enough, still the discovered model will be incorrect. Consider the Petri net from Figure 5.1. If we suppose that very often, task h is processed in 1 time unit, task g in 3 time units and i in 2 time units and h always finishes its execution before i starts, then we will see many times the sequence “a,f,h,i,g,k,l” and very seldom the sequence “a,f,g,i,h,k,l”. This means that although k is the direct successor of h , we will not find the connection between h and k . The explanation is that too few “h,k” occurrences are present in the log and we do not have enough support to say that h and k are directly connected. In the extreme case, none of the “h,k” occurrence will be present in the log, which will result into an incomplete log.
3. the log size is too small. Even if the log is complete and the priorities of task executions are perfectly balanced, the discovered model can be incorrect.

In this chapter we plan to test how the above mentioned log characteristics and other possible characteristics influence the accuracy of the discovered process models.

5.4 Discovering the log-based relations

Our method for discovering the process model from a log file is based on finding the log-based relations that can exist between tasks. For example, if a task is always followed by another task, it is likely that there is a causal relation between both tasks. In order to find the ordering relations between tasks, we use the dependency/frequency table.

5.4.1 The dependency/frequency table

The construction of a so-called dependency/frequency (D/F) table from the process log information is the starting point of our method and was first used in Weijters and Aalst [2001a]. An excerpt from the D/F table corresponding to the log generated by using the Petri net from Figure 5.1, is shown in Table 5.1. For each pair of tasks x and y , the following information is abstracted out of the process log: (i) the identifiers for tasks x and y , (ii) the overall frequency of task x (notation $|X|$ ¹), (iii) the overall frequency of task y $|Y|$, (iv) the frequency of task x directly preceded by another task y $|Y > X|$, (v) the frequency of task x directly succeeded by another task y $|X > Y|$, (vi) the frequency of x directly or indirectly preceded by another task y , but before the next appearance of x $|Y >>> X|$, (vii) the frequency of x directly or indirectly succeeded by another task y , but before the next appearance of x $|X >>> Y|$.

Table 5.1: An excerpt from the D/F table corresponding to the log generated by using the Petri net from Figure 5.1.

x	y	$ X $	$ Y $	$ Y > X $	$ X > Y $	$ Y >>> X $	$ X >>> Y $
a	f	1800	850	0	850	0	850
f	g	850	850	0	438	0	850
c	d	446	504	0	0	0	0
g	h	850	850	412	226	412	438
b	f	950	850	0	0	0	0
i	h	850	850	226	212	638	212

5.4.2 The log-based relations

Discovering a model from process logs involves determining the dependencies among tasks. Four types of event dependencies have been introduced in Cook and Wolf [1998a]: direct, sequential, conditional and concurrent dependence.

¹We use a capital letter when referring to the number of occurrences of some task.

We already introduced in Section 4.3 the notions of process trace and process log, where the process log was considered to be free of noise and in that case, the frequencies of specific traces were not used. In this chapter we are especially focusing on process logs that may contain noise, thus the use of trace frequencies is crucial for process discovery.

Let us reconsider the definitions of the log-based relations from Section 4.3, i.e. the Definition 13. To illustrate the above definitions, we focus on the process log from Table 5.2, corresponding to the Petri net from Figure 5.1. If there is no noise, there are three possible situations in which a pair of tasks can be:

1. tasks c and e are in sequence: then $c > e$, $e \not> c$, thus $c \rightarrow e$;
2. there is a choice between tasks b and f : then $b \not> f$, $f \not> b$, thus $b \# f$ (and $f \# b$);
3. tasks h and i are in parallel: then $h > i$, $i > h$, thus $h \parallel i$ (and $i \parallel h$).

Table 5.2: A workflow log example corresponding to the Petri net from Figure 5.1.

Case number	Executed tasks
Case 1	a f g h i k l
Case 2	a b c e j l
Case 3	a f h g i k l
Case 4	a f g i h k l
Case 5	a b c e j l
Case 6	a b d j l
Case 7	a b c e j l

However, in case of noise, the notions presented in Definition 13 are not useful anymore. If we want to investigate the relation between c and e , we find that $c > e$. However, because of some noisy sequences, we may see also that $e > c$. Applying the Definition 13, we could conclude that tasks c and e are parallel, which is wrong, because they are actually in a causal relation. Similarly, looking at tasks b and f , it can happen that $b > f$ and $f > b$, because of noise. Investigating the relation between h and i , we can see that $h > i$ and $i > h$, in situations with and without noise.

Suppose now that we are aware of the existence of noise in a process log (which is a realistic assumption) and for two generic tasks x and y we have $x > y$ and $y > x$, (for example, $|X > Y|=10$ and $|Y > X|=2$). What is the relation between x and y : causal, exclusive or parallel? We have to find a way to distinguish between these relations, taking into account that noise can exist.

We plan to induce models that can be used to detect the relations between tasks. Next, we use the α process discovery algorithm introduced in Section 4.3.1 that constructs a process model using the Petri net formalism. The α algorithm considers first all tasks that stand in a causal relation. Then for all tasks that share locally the same input (or output) task, the exclusive/parallel relations are included to build the Petri net. Based on the choice for the α algorithm to build the Petri net, we plan to induce models that adopts its sequence of actions: first detect the causal relations,

and then determine the exclusive/parallel relations for all tasks that share the same local input (or output) task. In Section 5.5 we present the procedure to induce such models. Before we induce these models, first we have to develop some useful relational metrics, which are presented in the next subsection.

5.4.3 The relational metrics

The information contained in the D/F table is the basic material that we use to induce predictive models for detecting the log-based relations. However, the raw frequencies of the D/F table cannot be used directly as input features for inducing the rule-set. Rather, we have to develop useful relational metrics from these raw data that can be used as input features.

When thinking about good measures that can be used to detect the causal relation $x \rightarrow y$ between tasks x and y , we noticed that the frequencies $|X > Y|$ and $|Y > X|$ from the D/F table are important to predict the causal relation. Namely, when the difference between $|X > Y|$ and $|Y > X|$ is large enough, there is a high probability that x causes y . We develop three different measures that use the difference between $|X > Y|$ and $|Y > X|$: the causality metric CM , the local metric LM and the global metric GM .

$|X > Y|$ and $|Y > X|$ frequencies are also useful to detect exclusive/parallel relations. If both frequencies are zero or very small numbers, then it is likely that x and y are in an exclusive relation, while if they are both sufficiently high, then it is likely that x and y are in parallel relation. Therefore, we construct the metrics YX and XY which are obtained by dividing the frequencies $|X > Y|$ and $|Y > X|$ with the minimum of $|X|$ and $|Y|$.

The causality metric CM

The causality metric CM was first introduced in Weijters and Aalst [2001a]. If for a given process log it is true that when task x occurs, shortly later task y also occurs, it is possible that task x causes the occurrence of task y . The CM metric is computed as follows: if task y occurs after task x and n is the number of tasks between x and y , then CM is incremented with a factor $(\delta)^n$, where δ is a causality factor, $\delta \in [0.0, 1.0]$. We set $\delta = 0.8$. The contribution to CM is maximally 1, if task y appears right after task x and consequently $n = 0$. Conversely, if task x occurs after task y and again the number of tasks between x and y is n , CM is decreased with $(\delta)^n$. After processing the whole log, CM is divided by the minimum of the overall frequency of x and y .

The local metric LM

Considering tasks x and y , the local metric LM is expressing the tendency of causality of the succession relation $x > y$, by comparing the magnitude of $|X > Y|$ versus $|Y > X|$.

The formula for the local metric LM is:

$$LM = P - 1.96 \sqrt{\frac{P(1-P)}{N+1}}, P = \frac{|X > Y|}{N+1}, N = |X > Y| + |Y > X| \quad (5.1)$$

The idea of this measure is borrowed from statistics and it is used to calculate the confidence intervals for errors. For more details, see Mitchell [1995]. In our case, we are interested to know with a probability of 95% the likelihood of causality relation, by comparing the magnitude of $|X > Y|$ versus $|Y > X|$. For example, if $|A > B| = 30$, $|B > A| = 1$ and $|A > C| = 60$, $|C > A| = 2$, what is the most likely: a causes b or a causes c ? Although both ratios $\frac{|A>B|}{|B>A|}$ and $\frac{|A>C|}{|C>A|}$ equal 30, a is more likely to cause c than b . Our LM measure for tasks a and b gives a value of $LM = 0.85$ and for tasks a and c gives a value of $LM = 0.90$, which is in line with our intuition.

Let's now consider again the Petri net from Figure 5.1. If we suppose that the number of lines in the log corresponding to this Petri net is equal to 1000 (i.e. $\#L=1000$), we can have the following three situations:

1. $|C > E|=1000$, $|E > C|=0$, $LM=0.997$,
2. $|H > G|=600$, $|G > H|=400$, $LM=0.569$,
3. $|F > B|=0$, $|B > F|=0$, $LM=0$.

In the sequential case (situation 1), because e always succeeds c , $LM \cong 1$. When h and g are in parallel, in situation 2, $LM = 0.569$, i.e. a value much smaller than 1. In the case of choice between f and b , in situation 3, $LM = 0$. In general, we can conclude that the LM measure has a value close to 1 when there is a clear tendency of causality between tasks x and y . When the LM measure is close to 0, there is no causality relation between tasks x and y . When the LM measure has a value close to 0.5, then $x > y$ and $y > x$, but a clear tendency of causality cannot be identified.

The global metric GM

The previous measure LM was expressing the tendency of causality by comparing the magnitude of $|X > Y|$ versus $|Y > X|$ at a local level. Let us now consider that the number of lines in our log is $\#L=1000$ and the frequencies of tasks a , b and c are $|A|=1000$, $|B|=1000$ and $|C|=1000$. We also know that $|A > B| = 900$, $|B > A| = 0$ and $|A > C| = 50$ and $|C > A| = 0$. The question is: is a the most likely cause of b or c ? For a causes b , $LM = 0.996$ and for a causes c , $LM = 0.942$, so we can conclude that a causes both b and c . However, one can argue that c succeeds a less frequently, thus a should be considered the cause of b , and not of c .

Therefore, we build a second measure, the global metric GM :

$$GM = ((A > B) - (B > A)) \frac{\#L}{(A) * (B)} \quad (5.2)$$

The values for the GM and LM metrics are given in Table 5.3.

Table 5.3: Illustration of GM and LM measures.

X	No. of tasks	$ X > A $	$ A > X $	LM	GM
B	1000	0	900	0.99	0.90
C	1000	0	50	0.94	0.05

In conclusion, for determining the likelihood of causality between two tasks x and y , the GM metric is indeed a global metric because it takes into account the overall frequencies of tasks x and y , while the LM metric is a local metric because it takes into account only the magnitude of $|X > Y|$ versus $|Y > X|$.

The YX and XY metrics

The three metrics presented before were especially developed to be used as predictors for the causality relation, but they are not very useful for deciding between exclusive and parallel relations. However, $|X > Y|$ and $|Y > X|$ frequencies can be used again to decide between exclusive and parallel relations. When between x and y there is an exclusive relation, both $|X > Y|$ and $|Y > X|$ frequencies should be zero or a small value, while for the parallel case, both should be relatively high. Because the models that will be induced using these metrics as predictors must be general, we have to take into account also the frequencies of tasks x and y . Therefore we divide $|X > Y|$ and $|Y > X|$ with the minimum of $|X|$ and $|Y|$.

Thus, YX and XY are defined as follows:

- YX : the proportion of $|Y > X|$ divided by the minimum frequency of x and y
i.e. $YX = \frac{|Y > X|}{\min\{|X|, |Y|\}}$;
- XY : the proportion of $|X > Y|$ divided by the minimum frequency of x and y
i.e. $XY = \frac{|X > Y|}{\min\{|X|, |Y|\}}$;

In Table 5.4 and 5.6 are presented the values for the relational metrics of some task pairs, in case of a log generated by the Petri net shown in Figure 5.1. In the next section we show how these metrics can be used to induce models for predicting the log-based relations.

5.5 Inducing models for discovering log-based relations

The relational metrics presented so far are the basis of the models that we plan to induce in order to detect the log-based relations. Next, we want to use the α algorithm introduced in Section 4.3.1 that constructs a process model using the Petri net formalism. We want to induce two types of models: (i) a logistic regression model and (ii) rule-based model. Our plan is to focus on the following actions:

- induce a logistic regression model to predict the causal relation,
- induce a rule-based model to predict the causal relation,
- identify the model with the best performance,
- use this model to induce a second predictive model, that predicts exclusive/parallel relations.

The two induced models, i.e., the model that predicts causal relations and the model that predicts exclusive/parallel relations will conduce to the *three-step* method for detecting log-based relations that:

1. detects the causal relations,
2. determines the exclusive/parallel relations for all tasks that share the same local input (or output) task,
3. use the α algorithm to build the Petri net process model.

In Weijters and Aalst [2001a] a threshold value have been used to decide whether a causal relation exists between two tasks. However, in case of very noisy data, the rules used in this work were missing some causal relations. Our idea was to improve the method used in Weijters and Aalst [2001a] in two ways: (i) by using a combination of more relational metrics (ii) by inducing a model from experimental data, where log characteristics are varied. We kept the idea of using one global threshold value from Weijters and Aalst [2001a], i.e. we want to set a threshold value σ that is used to decide whether there is causal relation between two tasks or not.

We find the logistic regression a good candidate for our problem (see Section 2.2.2). The logistic regression is a form of regression used when the dependent is a dichotomy and the independent are continuous variables, categorical variables or both. The dichotomic characteristic that we want to predict is whether between tasks x and y there is a causal relation, which can be True or False.

The learning material used to induce the models are the generated logs, as described in Section 5.3. Namely, for each of the 400 log files (that resulted by considering noise, imbalance, etc.), a D/F table is built and finally all 400 separate D/F tables are combined into one big file, called the Experimental D/F file (ExpD/F).

We presented the induction of the logistic regression model to predict causal relations in Mărușter et al. [2002b]. We found out that using one global threshold has the drawback of being too rigid, e.g., some real causal relations are missed and false causal relations may be considered as causal relations. Therefore, we induced a rule-based model as an alternative approach to the logistic regression (Mărușter et al. [2003]). The rule-based model show a better performance for predicting causal relation, therefore we used it to induce the second predictive model, that predicts exclusive/parallel relations.

In Section 5.5.1 we describe in detail the induction of the logistic regression model. The induction of the rule-based model and the three-step method for detecting log-based relations are presented in Section 5.5.2. We evaluate the models in Section 5.5.3.

5.5.1 The logistic regression model

We want to develop a model that can be used to determine when two tasks x and y are in causal relation. The idea is to combine the CM , LM and GM metrics described earlier and to find a threshold value σ over which two tasks x and y can be considered to be in causal relation.

The logistic regression estimates the probability of a certain characteristic occurring. We want to predict whether “tasks x and y are in causal relation”, that can be True/False. Therefore, we set as dependent variable the C field from the ExpD/F file, that is marked with a “T” or “F”. In Table 5.4 is shown an excerpt from the ExpD/F

table with the corresponding relational metrics, for a log generated using the Petri net from Figure 5.1.

Logistic regression transforms the dependent variable into a *logit* variable, i.e., the natural log of the odds of the dependent occurring or not. For example, if the dependent is measuring the quality 'success' or 'failure' of a certain event, and if the proportion of success is 0.2 in the data, the odds equal 0.25 ($0.2/0.8=0.25$). In this way, logistic regression estimates the probability of a certain event's quality occurring. Note that logistic regression calculates changes in the log odds of the dependent, not changes in the dependent itself, as linear regression does.

The independent variables used to induce the logistic regression model are the three metrics that we built earlier, i.e. the causality metric *CM*, the global metric *GM* and the local metric *LM*. In conclusion, we want to obtain a model that, for two tasks x and y , will estimate the probability $\hat{\pi}$ of x and y being in causal relation. Given a certain threshold value σ , we will decide that x and y are in causal relation if $\hat{\pi} \geq \sigma$.

Table 5.4: An excerpt from the ExpD/F table with the corresponding relational metrics. The column “*C*” contains a “T” or a “F”, whether between tasks x and y there is a causal relation or not.

x	y	$ X $	$ Y $	$ Y > X $	$ X > Y $	$ Y \gg X $	$ X \gg Y $	<i>CM</i>	<i>GM</i>	<i>LM</i>	<i>C</i>
a	f	1800	850	0	850	0	850	1.00	1.00	0.99	T
f	g	850	850	0	438	0	850	0.90	1.09	0.99	T
c	d	446	504	0	0	0	0	0.00	0.00	0.00	F
g	h	850	850	412	226	412	438	-0.01	-0.43	0.31	F
b	f	950	850	0	0	0	0	0.00	0.00	0.00	F
i	h	850	850	226	212	638	212	-0.40	-0.03	0.43	F

The general form of our logistic regression model is

$$\log \frac{\pi}{(1-\pi)} = B_0 + B_1 * LM + B_2 * GM + B_3 * CM \quad (5.3)$$

where the ratio $\frac{\pi}{(1-\pi)}$ represents the odds. The significance of individual logistic regression coefficients B_i is given by the Wald statistics which indicates significance in our model; that means that all independent variables have a significant effect on predicting the causal relation (Wald tests the null hypothesis that a particular coefficient B_i is zero).

The model goodness of fit test is a Chi-square function that tests the null hypothesis that none of the independents are linearly related to the log odds of the dependent. Inducing the model, we obtain a value of 108262.186, at probability $p < .000$, inferring that at least one of the population coefficients differs from zero. The coefficients of the logistic regression model are shown in Table 5.5.

Using the B_i coefficients from Table 5.5, we can write the Logistic Regression

Table 5.5: Logistic analysis summary: the discrete dependent variable measures the question “are tasks x and y in causal relation?”, and the used predictors are the relational metrics CM , GM and LM .

Variables in the Equation	B_i	Wald	df	Sig ^a	Exp(B)
CM	8.654	4490.230	1	0.000	5735.643
GM	4.324	920.638	1	0.000	75.507
LM	6.376	2422.070	1	0.000	587.389
Constant	-8.280	4561.956	1	0.000	0.000

^a means significant at $p < 0.01$

expression LR from Equation 5.4:

$$LR = -8.280 + 6.376 * LM + 4.324 * GM + 8.654 * CM \quad (5.4)$$

Thus, the estimated probability can be calculated with the formula shown in Equation 5.6:

$$\hat{\pi} = \frac{e^{LR}}{1 + e^{LR}} \quad (5.5)$$

The influence of LM , GM and CM can be detected by looking at column $Exp(B)$ in Table 5.5. For example, when CM increases one unit, the odds that the dependent=1 increase by a factor of 5736, when the others variables are controlled. Comparing between GM , LM and CM , we can notice that CM is the most important variable in the model.

Looking at correct and incorrect estimates we can inspect the model performance. Our model predicts the “T” value of the dependent in 95,1% of cases and the “F” value of the dependent in 99,2% cases. These values for correct/incorrect estimates are obtained at a cut value of 0.8, i.e. are counted as correct estimates those values that exceed the value 0.8. We could set this cut value at, let’s say 0.5, but we are interested in knowing the classification score when the estimated probability is high. Because 95% of the tasks which are in causal relation are correctly predicted by the model, we conclude that the threshold $\sigma = 0.8$ is appropriate. This means that we decide to be a causal relation between tasks x and y , if the estimated probability $\hat{\pi} \geq 0.8$.

Let us illustrate these with an example, considering the Table 5.4. Suppose that we question if task a and f are in causal relation. We have to estimate the probability $\hat{\pi}$ by applying the formula from Equation 5.6, i.e.

$$\hat{\pi} = \frac{e^{-8.280+6.376*0.99+4.324*1.00+8.654*1.00}}{1 + e^{-8.280+6.376*0.99+4.324*1.00+8.654*1.00}} = \frac{e^{11.01}}{1 + e^{11.01}} = 0.999 \quad (5.6)$$

Clearly, $\hat{\pi} = 0.999 \geq 0.8$, thus we can conclude that a and f are in causal relation.

In the next section, we describe an alternative model, i.e. the induction of a rule-based model.

5.5.2 The rule-based model

As an alternative to the logistic regression model, we want to induce a rule-based model for detecting log-based relations. Our intuition is that a model that does not require the existence of one global threshold would act more flexibly and subsequently, will result into a better performance. We intend to develop a *three-step method*, that (i) detects the causal relations, (ii) for all tasks that share the same local input (or output) task, determines the exclusive/parallel relations and (iii) applies the α algorithm to build the Petri net process model.

In Section 5.4.3 we introduced five relational metrics CM , GM , LM , YX and XY to be used in determining the causal and exclusive/parallel relations. The idea is to use the learning material generated in Section 5.3, to compute the relational metrics and to induce decision rule-sets that detect the relations between two tasks.

When choosing a learning algorithm, we have to establish some criteria. First, we want to obtain a model that can be easily understood and second, we are interested in a fast and efficient algorithm. Ripper is an algorithm that induces rule-sets (Cohen [1995]). It has been shown that Ripper is competitive with the alternative algorithm C4.5rules in terms of error rates, but more efficient than C4.5rules on noisy data (Cohen [1995]), thus it seems to meet our requirements.

For inducing a rule-set, we have to provide a set of examples, each of which has been labelled with a *class*. In our case, we have four possible classes which are the types of log-based relations that can exist between two tasks: “c” for causal, “e” for exclusive, “p” for parallel and “i” for an inverse causal relation. However, we are interested to induce rule-sets for detecting the first three relations, i.e. “c”, “e” and “p” (the “i” relation is not interesting, because it is not used by the α algorithm to construct the Petri net).

Because we have to induce two independent rule-sets, we need to separate the learning material needed in the first step, from the learning material needed in the second step. Detecting the causal relations is the first step, thus we label each instance of the generated learning material with a “c”, whether there is a causal relation between the tasks and with an “n” if not. In the second step, we select only those pairs of tasks which share the same cause or the same direct successor task. We label these instances with an “e” or a “p”, whenever between the tasks there is an exclusive or a parallel relation.

An excerpt of the table with the class labelling is presented in Table 5.6. Note the pairs (c,d) and (g,h) which are labelled in Step 1 with an “n” (in the first step they are used as non-causal examples), while in Step 2 they are labelled “e” and “p” respectively, being selected to induce rules that distinguish between the exclusive and the parallel relation.

The induction of the two rule-sets is described in the following two subsections.

The induction of the rule-set for detecting causal relations

The computed relational measures corresponding to the 400 generated logs are stored in one file that serves as training material for the induction of the rule-sets. In order to obtain these rule-sets, we use Ripper algorithm (Cohen [1995]). The Ripper algorithm produces ordered rules, by using different methods. We use the default

Table 5.6: Excerpt from the learning materials used to induce the rule-set for detecting in Step 1 the causal relations and in Step 2, the exclusive/parallel relations, using the log generated by the Petri net presented in Figure 5.1. x and y represent the task identifiers, CM , GM , LM , YX and XY are the calculated relational measures, and “ Rel ” contains the “c”, “e” and “p” letter to label the pairs in causal, exclusive and parallel relations.

Step	x	y	CM	GM	LM	YX	XY	Rel
1	a	f	1.000	1.000	0.998	0.000	1.000	c
1	a	b	1.000	1.000	0.998	0.000	1.000	c
1	f	g	0.903	1.091	0.996	0.000	0.515	c
1	f	h	0.857	1.026	0.995	0.000	0.485	c
1	b	a	-1.000	-1.000	0.000	1.000	0.000	n
1	c	d	0.000	0.000	0.000	0.000	0.000	n
1	g	h	-0.019	-0.436	0.317	0.485	0.266	n
2	b	f	0.000	0.000	0.000	0.000	0.000	e
2	c	d	0.000	0.000	0.000	0.000	0.000	e
2	g	h	-0.019	-0.436	0.317	0.485	0.266	p
2	i	h	-0.404	-0.035	0.437	0.266	0.249	p

method, i.e. order by increasing frequency. After arranging the classes, Ripper finds rules to separate class1 from classes class2, ..., classn, then rules to separate class2 from classes class3, ..., classn, and so on. To obtain a rule-set for detecting the causal relations, we use only the instances labelled with “c” or “n”. We obtain 33 ordered rules for class “c” (“n” is the default class); we refer this rule-set as RIPPER_CAUS. The training error rate for RIPPER_CAUS is 0.08% (the training error rate represents the rate of incorrect predictions made by the model over the training data set). Because the feature LM appears multiple times in several rules, we simplify these rules by considering the intersection of the intervals specified by the LM metric. Below are shown the rules with a coverage of over 100 positive instances and less than 7 negative instances. We can remark that these rules cover quite a lot of positive instances and have few negative counterexamples.

Rule1: IF $LM \geq 0.949$ AND $XY \geq 0.081$ THEN class c [10797 pos, 0 neg]

Rule2: IF $LM \geq 0.865$ AND $YX = 0$ AND $GM \geq 0.224$ THEN class c [1928 pos, 6 neg]

Rule3: IF $LM \geq 0.844$ AND $CM \geq 0.214$, $CM \leq 0.438$ THEN class c [525 pos, 1 neg]

Rule4: IF $LM \geq 0.741$ AND $GM \geq 0.136$ AND $YX \leq 0.009$ AND

$CM \geq 0.267$ AND $CM \leq 0.59$ THEN class c [337 pos, 0 neg]

Rule5: IF $XY \geq 0.6$ AND $CM \leq 0.827$ THEN class c [536 pos, 0 neg]

Rule6: IF $LM \geq 0.702$ AND $YX \leq 0.009$ AND $GM \geq 0.36$ THEN class c [273 pos, 0 neg]

Rule7: IF $LM \geq 0.812$ AND $CM \leq 0.96$ AND $GM \geq 0.461$ THEN class c [142 pos, 0 neg]

Let us interpret these rules. Suppose that we want to detect the relation between two tasks x and y . Rule1 has the highest coverage of positive examples, i.e. almost 70% of “c” instances match this rule. E.g., if the LM measure has a very high value

(i.e. there is a big difference in magnitude between $|X > Y|$ and $|Y > X|$ frequencies) and the XY measure is exceeding a small value, there is a high chance to be a causal relation between x and y . The first condition of Rule2 specifies LM to be high. The second condition specifies that YX measure must be 0, i.e. $|Y > X| = 0$. The third condition requires the global measure GM to exceed 0.2, i.e. the difference between $|X > Y|$ and $|Y > X|$ frequencies, accounted by the overall frequencies of x and y should be sufficiently high. In general, the rules require the LM measure to exceed a high value, YX to be a value close to zero, while XY should be bigger than 0. Also, CM and GM measures should be sufficient large.

The conclusion is that we successfully developed (i) measures that are useful to predict causal relations and (ii) the rule-set RIPPER.CAUS seems to have a high performance. In Section 5.5.3 we provide further evaluation of our model.

The induction of the rule-set for detecting exclusive/parallel relations

In order to induce the rule-set for detecting exclusive/parallel relations, from the whole material generated in Section 5.3, we select only the pairs of tasks which share the same cause or the same direct successor task. In Table 5.6, at Step 2, the pairs of tasks in exclusive and parallel relations and the corresponding relational measures are shown. We see that tasks g and h have as same common cause the task f , and tasks b and f have as same common cause the tasks a . The pairs in exclusive relation are labelled with “e” (e.g. the pair of tasks (b, f)) and those in parallel relations with “p” (e.g. the pair (g, h)).

When inducing the rule-set for detecting causal relations, we were primarily interested in rules that predict the “c” class. Here we want to develop rules for both exclusive and parallel relations (“e” and “p” classes) and to inspect the difference (if any). We run Ripper algorithm with the method to produce unordered rules: Ripper will separate each class from the remaining classes, thus ending up with rules for every class. Conflicts are resolved by deciding in favor of the rule with lowest training-set error. We obtain the RIPPER_ANDOR rule-set with 15 unordered rules, 7 for class “e” and 8 for class “p”, with the training error rate 0.38%.

The 14 unordered rules are the following (we omit one rule with very low coverage):

- Rule1: IF $XY=0$ AND $GM \geq 0$ THEN class e [4734 pos, 32 neg]
- Rule2: IF $XY \leq 0.01$ AND $CM \leq -0.35$ AND $YX \leq 0.04$ THEN class e [486 pos, 0 neg]
- Rule3: IF $YX \leq 0.01$ AND $LM \leq 0.31$ AND $CM \geq -0.02$ AND $CM \leq 0.04$ THEN class e [3006 pos, 2 neg]
- Rule4: IF $YX \leq 0.01$ AND $CM \leq -0.26$ THEN class e [588 pos, 8 neg]
- Rule5: IF $YX \leq 0.01$ AND $XY \leq 0$ AND $CM \geq -0.06$ AND $CM \leq 0.01$ THEN class e [2704 pos, 7 neg]
- Rule6: IF $XY \leq 0.01$ AND $CM \geq 0.29$ THEN class e [253 pos, 0 neg]
- Rule7: IF $XY \geq 0.01$ AND $YX \geq 0.02$ THEN class p [5146 pos, 0 neg]
- Rule8: IF $XY \geq 0.02$ AND $CM \geq -0.24$ AND $LM \geq 0.33$ THEN class p [3153 pos, 0 neg]
- Rule9: IF $YX \geq 0.01$ AND $CM \geq -0.26$ AND $CM \leq -0.07$ THEN class p [1833 pos, 1 neg]
- Rule10: IF $XY \geq 0.01$ AND $CM \geq -0.24$ AND $CM \leq -0.04$ THEN class p [2227 pos, 3 neg]
- Rule11: IF $YX \geq 0.01$ AND $CM \geq 0.06$ THEN class p [1523 pos, 1 neg]
- Rule12: IF $GM \leq -0.01$ AND $CM \geq 0.08$ THEN class p [223 pos, 0 neg]
- Rule13: IF $YX \geq 0.02$ AND $GM \leq -0.03$ THEN class p [1716 pos, 1 neg]

Rule14: IF $XY \geq 0.06$ THEN class p [3865 pos, 0 neg]

Let us inspect first the RIPPER_ANDOR rule-set for class “p”. First, Rule7, which has the highest coverage (it matches almost 93% from “p” instances), requires that both XY and YX measures to exceed zero, what we actually expected: if there are sufficient occurrences of task x and task y next to each other, then there is likely to be a parallel relation between them; if there are few such occurrences, it is likely to be some noise involved, and then the relation between tasks is the exclusive one. Rule14 goes in the same direction as Rule7, but requires only the measure XY to be higher than zero. The rest of rules for class “p” have also high coverage; differently from Rule7 and Rule14, they include combinations of all five measures. For example, Rule8 specifies three conditions: the first one requires XY to be higher than zero. The second condition that involves the CM measure is less easy to interpret in the context of Rule8. The third condition specifies LM to be higher than 0.33: a value for LM that has to exceed 0.33 means that the difference between $|X > Y|$ and $|Y > X|$ frequencies should be relatively small, which is understandable in case of parallel tasks.

Looking to the rules for class “e”, we expect to have complementary conditions. Rule1 has the highest coverage, but has also 32 counterexamples. This rule specifies that XY should be zero and $GM \geq 0$, which makes sense: in case of choice between tasks x and y , we should not see any occurrence of x and y next to each other, which indeed leads to $XY=0$ and $GM=0$. In the rest of rules for class “e”, we see that mostly of time XY and YX should be smaller than 0.01, that ensures the detection of an exclusive relation when there is noise.

The involvement of the CM measure becomes clearer when inspecting all rules, both for the “e” and the “p” class. In general, in case of class “e”, CM should be found inside an interval around zero (Rule3 and Rule5), while in case of “p” class, CM should not reach zero (Rule9 and Rule10). Rule6 and Rule11 specify both that CM should be bigger than zero; the decision to be an exclusive or a parallel relation is based on the XY measure (Rule3), that should be smaller than 0.01, and on YX (Rule11) that should be bigger than 0.01. If there is a choice between tasks x and y and there exist cycles, then x and y do not appear next to each other (rather, y appears somewhere later after x), so the CM measure has to exceed a certain value, as in Rule6.

We can conclude that the obtained rules (i) make a good distinction between exclusive and parallel relations and (ii) the rule-set seems to have a good performance.

5.5.3 Evaluating the induced models

We evaluate our induced models by performing (i) 10-fold cross-validation on test material extracted from the learning material and (ii) model check on a completely new test material.

To evaluate our models, we use three performance indicators: the error rate, precision and recall. Error rate is not always an adequate performance measure, because it gives skewed estimates of generalization accuracy when classes are imbalanced in their frequency. When identifying the relations between tasks, we are interested to see an aggregate of the cost of false positives and false negatives, expressed in terms

of recall and precision. In case of causal relations, false positives are false causal relations found, i.e. linking tasks which are not causally related. False negative are actual causal relations that are omitted from the Petri net. We use the F-measure, that combines precision and recall (see Equation 2.3).

10-fold cross-validation test

Performing 10-fold cv experiments with the two rule-based models (i.e. the rule-set for detecting causal relations and the rule-set for detecting exclusive/parallel relations), we calculate the average value of the three performance indicators, i.e., error rate, precision and recall. We use the same 10 samples from the 10-fold cv experiments to induce 10 logistic regression models. In case of the rule-based models, we obtain for class “c” an average error rate of 0.11%, 99.35 precision, 98.09 recall and 98.72 F-measure. In case of logistic regression models, we obtain for class “c” an average error rate of 2.70%, 99.10 precision, 97.52 recall and 98.29 F-measure.

Detecting the “c” class, the performance indicators of the logistic regression model are compared with the performance indicators of the rule-set model, by performing a *paired t-test*. The paired t-test compares the means of two variables for a single group. It computes the differences between values of the two variables for each case and tests whether the average differs from 0. In our case, one variable contains the error rates for the 10-fold cross-validation experiments using the rule-based model, and the second variable contains the error rates for the 10-fold cross-validation experiments using the logistic regression model. We could use the paired t-test, because the materials employed in the 10-fold cross-validation experiments are the same randomly extracted samples from the whole population of examples.

The outcome of the paired t-test is that there is a significant difference between the performance (expressed in error rates) of the rule-based model and of the logistic regression model. Namely, we obtain the mean of the difference equal with 2.601 and the standard error of the mean equal with 0.038. This results into a t value of 67.26, at a probability $p < 0.05$, thus we can reject the null hypothesis that the population mean of difference scores is 0. We can conclude that the rule-based model significantly outperform the logistic regression model. Subsequently, we will prefer the rule-based model to detect “e” and “p” class. Detecting classes “e” and “p”, Ripper gets an averaged error rate of 0.46%. On class “e”, Ripper obtains 98.99 precision, 99.68 recall and 99.33 F-measure, while for class “p”, it gets 99.72 precision, 99.08 recall and 99.40 F-measure. In Table 5.7 are summarized the performance indicators of the 10-fold cv experiments performed so far.

Next, we measure the propagated error rate. So far, we inspected the performance of (i) the first rule-set for detecting causal relations and (ii) the second rule-set for detecting exclusive/parallel relations separately. When we induced the second rule-set, we initially selected all the task pairs that share a common cause or a common direct successor. This selection is made from “perfect” data, because we know which are the task pairs that share a common cause or a common direct successor in the learning material. It is interesting to check the performance of a model based on predicted data, i.e., to use the first model to predict the causal relations. From this new learning material, we select the task pairs that share a common cause or a common direct successor and we induce with Ripper a new rule-set that detects

Table 5.7: The averaged error rates, precision, recall and F-measure for the 10-fold cv experiments, for the Ripper rule-sets and for the logistic regression model.

<i>10-fold cv</i>	<i>error rate</i>	<i>precision</i>	<i>recall</i>	<i>F[0.1]</i>
Logistic regression “c” class	2.70%	99.10	97.52	98.29
Ripper “c” class	0.11%	99.35	98.09	98.72
Ripper “e” class	0.46%	98.99	99.68	99.33
Ripper “p” class	0.46%	99.72	99.08	99.40

exclusive/parallel relations. The 10-fold averaged error rate of this new second rule-set is 0.36% and the averaged F-measure for “e” and “p” classes is 99.83 and 99.85, respectively. These performance indicators are comparable with the performance indicators of the first rule-set induced from perfect data (the averaged error rate is 0.46% and the F-measure is 99.33 for “e” and 99.40 for “p” classes). Because the performance indicators do not differ significantly, we have enough support to use for future predictions the first rule-set for detecting causal relations, induced from perfect data.

Based on the 10-fold cross validations experiments, we can say that both rule-sets (i.e. the rule-set that detects causal relations and the rule-set that detects exclusive/parallel relations) have high performance on new data. However, this performance was checked on test data that are randomly extracted from the generated learning material. The learning (and testing) material used so far was generated based on a fixed set of Petri-nets.

Testing the model on new data

In order to check how robust our models are on predicting log-based relations using completely new data, we use a new Petri net structurally different from the Petri nets used to induce the rule-sets. We used the same methodology to generate log, by producing noise, imbalance and different log size, as presented in Section 5.3.

The error rate in case of predicting causal relations using the logistic regression model is 5.7%. Applying the rule-set RIPPER_CAUS on this new test material results in an error rate of 0.31%. We can note that the rule-set model RIPPER_CAUS performs better than the logistic regression model.

Applying the rule-set RIPPER_ANDOR that detects exclusive/parallel relations, it results in an error rate of 0.90%. The confusion matrix and the F-measure for the new test material by applying RIPPER_CAUS and RIPPER_ANDOR rule-sets are presented in Table 5.8. The performance indicators for both rule-sets in case of new test material are comparable with the two Ripper 10-folds experiments, although the performance indicators for the latter are slightly better (see Table 5.7 for comparison).

We can conclude that our two rule-sets show good performance also in case of new data, generated by a Petri net with a very different structure than the Petri nets used to induce the rule-sets.

The general conclusion is that both models show good performance. However, the rule-based model for detecting log-based relations significantly outperformed the

Table 5.8: The confusion matrix and performance results for the rule-sets RIPPER_CAUS and RIPPER_ANDOR on the new test data.

	<i>Predicted</i>			<i>Predicted</i>	
<i>Observed</i>	<i>c</i>	<i>n</i>	<i>Observed</i>	<i>e</i>	<i>p</i>
<i>c</i>	4246	254	<i>e</i>	1181	19
<i>n</i>	79	104321	<i>p</i>	0	900
<i>Recall</i>	94.36	99.92	<i>Recall</i>	98.42	100.00
<i>Precision</i>	98.17	99.76	<i>Precision</i>	100.00	97.93
<i>F</i> [0.1]	96.23	99.84	<i>F</i> [0.1]	99.20	98.96

logistic regression model. We connect this result with the fact that the rule-based model is more flexible than the method based on the logistic regression, which require a fixed threshold value. Subsequently, we are interested to investigate how log characteristics are influencing the performance of our two rule-based models.

5.6 The influence of log characteristics on the rule-based model performance

Finding the causal, exclusive and parallel relations with our method does not necessarily results in Petri nets equivalent with the original Petri nets used to generate the learning material. We already discussed in Section 4.3.2 for which class of Petri nets it is possible to rediscover the original net using the α algorithm, assuming log completeness and no noise in the process log. The method presented in this chapter provides a solution to construct the Petri net model from a process log when the log is incomplete and noisy. However, the degree of incompleteness and noise is affecting at a certain extent the quality of the discovered process model.

By generating experimental data where variations appear in the number of task types, imbalance, noise and log size, we attempt to control how our method misses or incorrectly predicts some relations. We are interested now to investigate the influence of these variations on the rule-sets performance.

In order to inspect the rule-sets performance when number of task types, imbalance, noise and log size are varied, we record the F-measure obtained by applying rule-sets RIPPER_CAUS and RIPPER_ANDOR on each of the 400 individual log files. Three types of F-measures can be calculated:

1. F_C: the F-measure obtained applying the rule-set RIPPER_CAUS. This F-measure is calculated with the formula from Equation 2.3, where TP are the number of task pairs in “c” relation classified as “c”, FP are the number of task pairs in “n” relation classified as “c” and FN are the number of task pairs in “c” relation classified as “n”.
2. F_E: the F-measure obtained with the rule-set RIPPER_ANDOR, without considering the propagated error. This means that the previous step of predicting causal relations is considered to execute without errors. The F_E measure is

calculated with the same formula from Equation 2.3, where TP are the number of task pairs in “e” relation classified as “e”, FP are the number of task pairs in “p” relation classified as “e” and FN are the number of task pairs in “e” relation classified as “p”.

3. F_E_PROP: the F-measure obtained with rule-set RIPPER_ANDOR, considering the propagated error. This means that in the previous step, some causal relations were missed or incorrectly found. The F_E_PROP is also calculated with formula from Equation 2.3. TP is the number of task pairs that meet the requirement to be in “e” relation, but this also includes the pairs which apparently have the same cause or the same direct successor (because some causal relation were incorrectly found in the previous step). FP is the number of task pairs in “p” relation classified as “e” and FN is the number of task pairs in “e” relation classified as “p”.

Similar formulas are used to compute the F_P and F_P_PROP for pairs of tasks in parallel relations.

We are interested to investigate how the number of task types, imbalance, noise and log size influence the prediction of causal and exclusive/parallel relations. We consider the averaged values of F_C, F_E_PROP and F_P_PROP for all 400 logs. The reason why we consider only F_E_PROP and F_P_PROP is that we are interested in the propagated performance, and not on a performance that assume perfect predictions in the previous step.

In Figure 5.2 a. we can see how the number of task types is influencing the averaged F_C. Note that the performance is dropping a little in case of the Petri net with 22 task types. A possible explanation is that this particular Petri net has a complex structure which is more difficult to be predicted. The same effect is depicted in Figure 5.2 b.

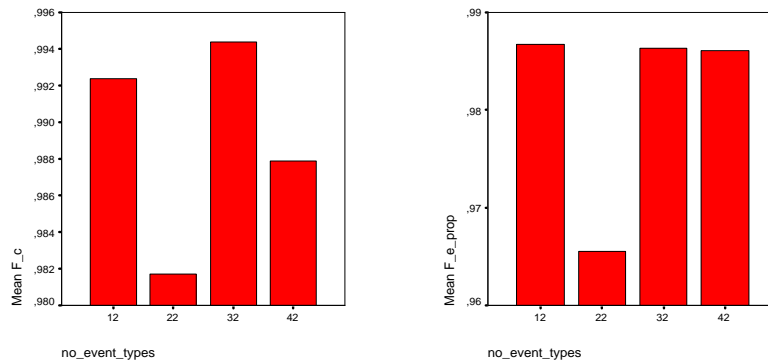
How imbalance in AND/OR splits affects the performance, is shown in Figure 5.3 a. Looking at F_C measure, we see that when the imbalance is increased, the performance is decreasing. A different situation is shown in Figure 5.3 b., where it seems that if the imbalance is increasing, the performance of finding exclusive relations is also increasing. It seems that a higher level of imbalance helps in distinguishing between exclusive and parallel relations. Inspecting the data, we remark that when the Petri nets are more balanced, than pairs in “e” relation are easier confused with pairs in “p” relation. A possible explanation can be that a rule for “p” class with a very high coverage often misclassifies “e” instances in certain conditions. Rule7 from the model RIPPER_ANDOR has the highest coverage, as we can see below:

```
Rule7: IF XY>=0.01 AND YX>=0.02 THEN class p [5146 pos, 0 neg]
```

When classifying “e” instances in case of balanced Petri nets, both XY and YX can exceed 0.01 and 0.02 (because both “x,y” and “y,x” can occur in the log with comparable probability), thus such instances will be incorrectly classified as “p”. When classifying “e” instances in case of unbalanced Petri nets, either XY will exceed 0.01, or YX will exceed 0.02, thus such instances have smaller chance to be classified as “p”.

The influence of noise on both performance measures F_C and F_E_PROP are presented respectively in Figure 5.4 a and b. They show the same expected behavior, i.e. if the noise is increasing, the performance is decreasing.

Figure 5.5 a and b illustrates how the performance measures F_C and F_E_PROP are influenced by the log size. As we expected, the incompleteness of the log is affecting the performance of finding causal relations: as log size increases, performance increases. However, as the log size increases, the performance of detecting exclusive relations decreases. Inspecting the data, we remark that when the log is larger, than pairs in “e” relation are sometimes easier confused with pairs in “p” relation. The explanation also relates Rule7. When classifying “e” instances in case of a complete log, both XY and YX can exceed 0.01 and 0.02 (because both “x,y” and “y,x” can occur in the log with comparable probability), thus such instances will be incorrectly classified as “p”. When classifying “e” instances in case of incomplete log, either XY will exceed 0.01, or YX will exceed 0.02, thus such instances have smaller chance to be classified as “p”.



a. No. of task types vs. F_C b. No. of task types vs.F_E_PROP

Figure 5.2: The effect of the number of task types on rule-set performance.

Based on the above findings, we can formulate the following conclusions:

- As expected, more noise, less balance and fewer cases, each have a negative effect on the quality of the results. The causal relations can be predicted more accurately if there is less noise, more balance and more cases.
- There is no clear evidence that the number of task types has an influence on the performance of predicting causal relations. However, causal relations in a structurally complex Petri net can be more difficult to detect.
- Because the detection of exclusive/parallel relations depends on the detection of the causal relations, it is difficult to formulate specific conclusions for the quality of exclusive/parallel relations. It appears that noise is affecting exclusive and parallel relations in a similar way as the causal relations, e.g., if the level of

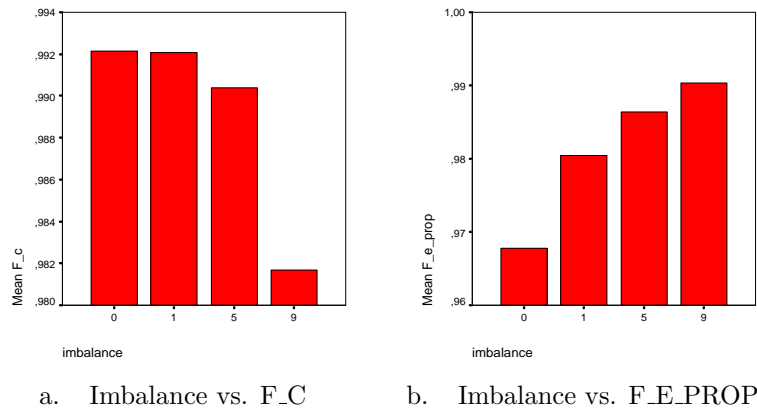


Figure 5.3: The effect of imbalance on rule-set performance.

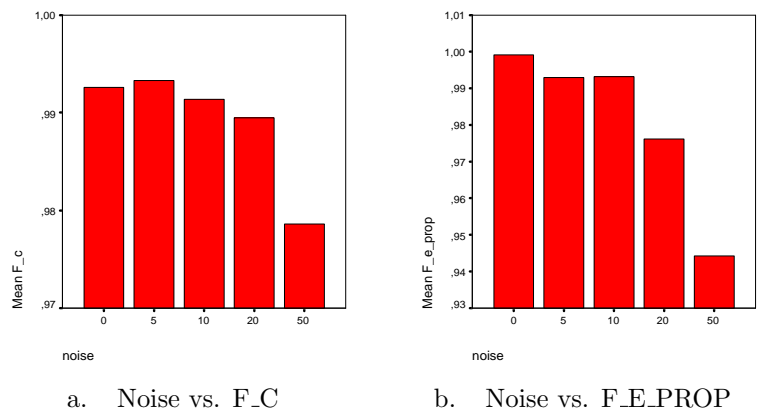


Figure 5.4: The effect of noise on rule-set performance.

noise is increasing, the accuracy of finding the exclusive/parallel relations is decreasing.

When discovering real process data, the above conclusions can play the role of useful recommendations. Usually it is difficult to know the level of noise and imbalance beforehand. However, during the discovery process it is possible to collect data about these metrics. This information can be used to motivate additional efforts to collect more data.

Finally, it is worthwhile to mention one aspect, namely to explain the reason why our rule-based model has such a good performance in classifying log-based relations. We think that one possible explanation resides in the way the experimental data was generated. We tried to vary as realistic as possible the process characteristics that may affect the process log (e.g. the total number of task types, the amount of available information in the process log, the amount of noise and the execution priorities in

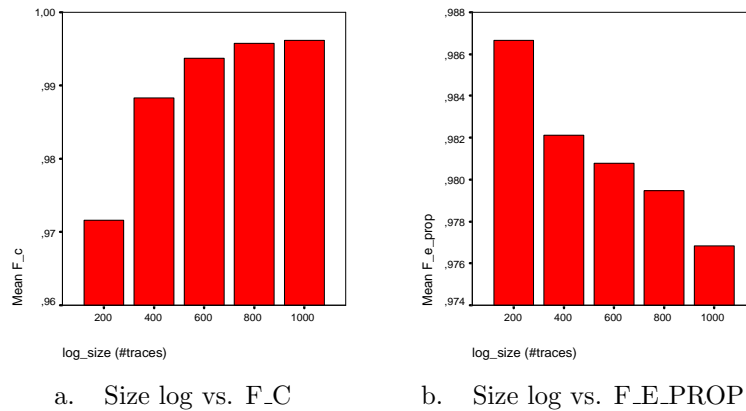


Figure 5.5: The effect of log size on rule-set performance.

OR-splits and AND-splits), but we may have missed some other characteristics that would negatively influence the model’s performance. Also, when we designed our Petri nets to generate the learning material, we can be (unintentionally!) responsible of a sort of “mannerism”. To prevent such a bias, we test our approach in Chapter 6 on simulated workflow data generated by other people and also on real data.

5.7 Conclusions

We developed a method that discovers the underlying process from a process log. We generated artificial experimental data by varying the number of task types, noise, execution imbalance and log size. Using these data we aimed to induce models with high accuracy on classifying new data.

Two types of models have been induced: a logistic regression model and a rule-based model. The rule-based model for detecting log-based relations significantly outperformed the logistic regression model. Namely, classifying causal relations, the rule-based model gets an average performance on the 10-fold cross-validation experiments of 99,89%, while the logistic regression gets an average performance of 97.3%. We connect this result with the fact that the rule-based model is more flexible than the method based on the logistic regression, which require a fixed threshold value.

We came to a three-step method: the first step employs the rule-based model to detect the causal relations; after the causal relations are found, the second rule-based model detects the exclusive/parallel relations between tasks that share the same cause or the same direct successor. Knowing the causal and exclusive/parallel relations and using the α algorithm presented in Section 4.3.1, the Petri net is built to obtain the process model.

Our rule-sets used in the three-step method have a very high performance in classifying new data, being able to find almost all relations in the presence of parallelism, imbalance and noise. Also, we tested our method on a process log generated by a more complex Petri net than the learning material, resulting in a performance close

to that on normal held-out test material.

Using the experimental data we investigated the influence of process log characteristics on our model performance. The causal relations can be predicted more accurately if there is less noise, more balance and more cases. However, causal relations in a structurally complex Petri net can be more difficult to detect. How process log characteristics are influencing the prediction of exclusive/parallel relations is less clear. It appears that noise is affecting exclusive and parallel relations in a similar way as the causal relations, e.g., if the level of noise is increasing, the accuracy of finding the exclusive/parallel relations is decreasing.

The current experimental setting confirmed some of our intuitions, e.g. that noise, imbalance and log size are factors that indeed affect the quality of the discovered model. However, in real processes more complex situations than we are aware of could be encountered. In Chapter 6 we test our practical approach of some simulated and real-world business data.

Chapter 6

Applications

In Chapters 4 and 5 we presented two approaches for discovering a process from sequence data. The issues presented in Chapter 4 provide results from a theoretical point of view, on discovering a process model from complete and noise-free process log. In Chapter 5, we provided a method that can discover, with high accuracy, the underlying process from noisy and incomplete data. We assessed the internal validity of our method in Section 5.5.3. Additionally, we are interested in the general applicability of our method, i.e., we want to check its external validity. To this purpose, we investigate to what extent our method results in process models that (i) reflect reality and (ii) conduce in meaningful insights into the considered process.

In this chapter we want to test the process discovery method presented in this thesis, on data resulting from (i) simulated processes and (ii) from real-world settings. In the first case, the aim is to test our approach on “non-biased” simulated data, i.e. data that are generated based on a different methodology than our methodology. In the second case, we are interested in checking to what degree the discovered process models reflects reality and provides useful insights into the considered process. We present three applications:

- we test our process discovery method on simulated workflow data generated by a business process management simulation tool, in the situation in which the process models are known.
- we test our process discovery method on real data of a governmental institution, resulted from the registration of an enterprize-specific information system.
- we test our process discovery method on real hospital data. We aim to discover the underlying process for each logistic patient group build in Chapter 3, from its corresponding log.

We already presented the evaluation of our discovery method using generated data. As we pointed out in Section 5.6, when we discussed the performance of our rule-based model to detect log relations, we have to eliminate the risk of assessing the validity of our approach based on possibly “biased” data. When we generated the data, we used a certain methodology, that could result into a certain bias. In order to eliminate this risk, it is important to perform evaluations on data sets different from

the material used to develop the method. In this sense, we employ simulated data that have been generated by other people, using a different methodology.

In most of the related research done in process discovery, the methods proposed have been evaluated also on simulated data (Herbst [2000a,b,c, 2001], Herbst and Karagiannis [1998, 1999, 2000], Schimm [2000a,b, 2001a,b, 2002], Weijters and Aalst [2001a,b, 2002]). The data are simulated given an initial process model which is used for data generation. By doing so, the discovered model can be compared with the initial process model. Sometimes, specific validation techniques are proposed to capture the correspondence between the execution of the designed process model and the execution of the discovered process model (Cook and Wolf [1998a], Cook and Wolf [1999]). However, it is interesting to check whether the proposed process mining technique is effective in more than “toy” problems.

When performing a case study on real data, we have to choose appropriate data. Not all data that can be collected from business records have an underlying process. For example, the products purchased in one day in a supermarket can show certain interesting patterns, but there is not an underlying *structured process*. A structured process assumes the existence of tasks that are executed in a certain order. Examples of structured processes are processes supported by the workflow management systems (Jablonski and Bussler [1996]).

There are different problems that can occur when performing evaluations on real world data: only some data are available (e.g., only those data that were perceived to be important are collected), data access can be problematic (due to confidentiality issues), etc. However, it is important to perform tests on real data, at least for two reasons: (i) to assess the external validity of the proposed method and (ii) to identify whether possible improvements need to be made to our method.

The structure of this chapter is as follows: in Section 6.1, the application of the discovery method on data simulated by ADONIS (a business process management tool) is described. The attempt of discovering a real-world process, namely the process of handling fines, is presented in Section 6.2. In Section 6.3 we discover the underlying process of the logistic multi-disciplinary patient groups, developed in Chapter 3. This chapter ends with some final conclusions.

6.1 Discovering business processes from simulated logs

In order to test our method with simulated data, we used workflow logs produced by the simulation component of the business process modelling tool ADONIS (Junginger et al. [2000]). ADONIS provides a process modelling language, as well as modelling support and interactive simulation of the modelled process. While simulating the process, all activities can be logged.

6.1.1 Data considerations

The ADONIS approach is intended to provide a tool for process modelling, interactive simulation and test case generation. When ADONIS is used for modelling purposes,

first the process activities need to be described. Second, ADONIS produces an executable specification by formalizing the textual parts previously described. This executable specification can be further analysed. To allow the user to inspect the resulting process model prototype, an interactive simulation is possible. ADONIS can simulate the process model by choosing random input and logs all relevant information. Further, these simulated logs can be used as a basis for testing the process model. More details about test case generation can be found in Kleiner and Herbst [2002].

The information recoded in the log refer to organizational and process aspects. In the organizational model, the workers and their roles are specified and in the process model, the processes, sub-processes and activities are defined.

The events recorded in ADONIS logs refer to the activities and the workers that perform the activities. These events can have the following types:

1. events relating activities
 - (a) sp: start process
 - (b) nt: new task
 - (c) ts: task sent
 - (d) ta: task arrives
 - (e) st: start task
 - (f) ft: finish task
 - (g) cp: complete process
 - (h) it: interrupt task
 - (i) ct: continue task
2. events relating workers
 - (a) pa: person arrives at work
 - (b) pl: person leaves

We will speak about *events* when referring to the eleven types of events previously mentioned, and about *tasks* when referring to activities to be executed in the process. A *case* is a particular instance of the process on which the task's action has an impact on.

To have a better understanding about ADONIS events, we use the Finite State Machine shown in Figure 6.1, that describes the life-cycle of tasks (i.e., all possible states of a task from creation to completion). State *New* is the state in which the task starts. Executing the event "send task", (i.e., sending the task to be processed), the task reaches in the state *Sent*. From this state, the event "task arrives" can be executed, i.e. the task becomes ready to be processed, in state *Arrived*. In this state, the task is typically in the worklist of one or more workers. From state *Arrived*, the task starts to be processed, it is removed from the worklist and subsequently, reaches in state *Active*. From state *Active*, the task can be finished or can be interrupted. If the task finishes, it reaches the final state *Completed*. If the task is interrupted (for

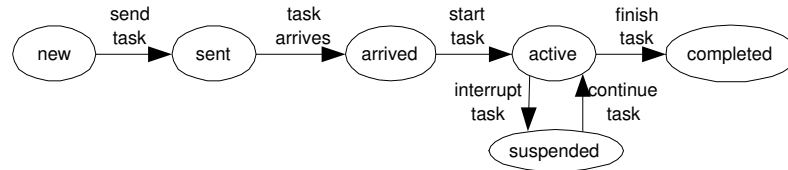


Figure 6.1: The life-cycle of tasks in ADONIS simulation module

Table 6.1: Excerpt from ADONIS log.

Record1	118903:nt:56:30:60:42
Record2	118903:nt:56:30:61:43
Record3	118903:ts:56:30:60:42:28:46
Record4	118903:ts:56:30:61:43:28:6
Record5	118903:ta:56:30:61:43:6
Record6	118903:st:56:30:61:43:6
Record7	118903:ta:56:30:60:42:46
Record8	119061:ft:56:28:60:41:29
Record9	119153:ft:56:10:58:11:16
Record10	119153:nt:56:10:60:44
Record11	119153:nt:56:10:61:45
Record12	119153:ts:56:10:60:44:16:13
Record13	119153:ts:56:10:61:45:16:27
Record14	119153:ta:56:10:60:44:13
Record15	119153:st:56:10:60:44:13
Record16	119153:ta:56:10:61:45:27
Record17	119153:st:56:10:61:45:27
Record18	119207:ft:56:12:58:13:53

example, because the worker leaves), it reaches in the state *Suspended*. Suspended tasks can move back to state *Active* via the event “continue task”.

In Table 6.1 we present an excerpt from an ADONIS log. The ADONIS logs comply with the following syntax:

`< time in seconds > : < event type > : < process id > : < process_instance id > : < task id > : < other parameters > .`

An important characteristic of ADONIS is the possibility to model tasks with multiple instances, which is also reflected in the log. In Table 6.1, at time 118903, for the process with the identifier “56” and instance “30”, seven events took place. Two new tasks, the task “60” with the instance “42” and task “61” with the instance “43”, have been issued. Also, the task “60” with instance “42” is sent by person “28” to person “46” and the task “61” with instance “43” is sent by person “28” to person “6”. In the same time, the task “61” with the instance “43” arrives at person “6”, the person “6” starts to work on task “61” with the instance “43”. Also, task “60” with the instance “42” arrives at person “46”.

There is a lot of information available in the ADONIS log. We have to decide which are the relevant event data that will form the process log, which we will use for discovering the underlying process model. Our discovery method handles tasks as atomic events, that are recorded into the process log as soon as they have been completed. In other words, the discovery method considers the events “st” (send task), “ta” (task arrives), “st” (start task) and “ft” (finish task) as one atomic action. However, ADONIS log contains records that refer specifically to the 11 already mentioned event types (9 for events relating activities and 2 for events relating workers). The problem is to choose those event types that characterize the dynamic behavior of the process in terms of sequencing of its major activities.

The events referring to workers, i.e. “pa” - person arrives at work - and “pl” - person leaves -, are not very useful for finding the process model, (rather they can be helpful to investigate how humans work). Also, event types “sp” (start process) and “cp” (complete process) are not interesting. The types of events that seem useful to grasp the dependencies between tasks are “nt” (new task), “ts” (task sent), “ta” (task arrives), “st” (start task) and “ft” (finish task). The idea is to select those records that belong to the same type of event and to form a process log for process discovery, by recording the task identifiers.

To illustrate this idea, from the data presented in Table 6.1, we can select those ADONIS records that share the same event type, process and process instance, as for example Record1, Record2 or Record3, Record4. In Table 6.2 is presented an excerpt from the process log that resulted by selecting all records that belong to the event type “finish task” from the ADONIS log.

Table 6.2: Example of process log traces resulting from ADONIS log.

```

58,61,60,63
58,61,60,63,65
58,61,60,63
58,60,61,63,65
58,60,61,63,58,61,60,63,58,61,60,63,65
58,61,60,63
58,61,60,63,58,61,60,63,58,60,61,63,58,61,60,63,58,60,61,63,58,60,61,63
58,61,60,63,58,61,60,63,58,60,61,63
58,60,61,63,58,61,60,63,58,61,60,63,58,61,60,63

```

The interesting problems are to investigate whether (i) given a selection of records belonging to a certain event type, are there relevant differences between the discovered models and (ii) if specific event type selections give better results than others. In the next subsection we try to answer to these questions.

Finally, we want to add that the traces presented in Table 6.2 are represented in text format, that can be handled by our discovery method. However, there have been efforts directed towards a common XML format for process logs (van der Aalst et al. [2003], van der Aalst and van Dongen [2002]). The problem is that each workflow management systems (ERP, CRM) have its own way of recording events. Neverthe-

less, once agreed upon a common format, it is fairly simple to extract information from enterprise-specific information systems and translate this to the XML format (as long as the information is there).

6.1.2 An application to a product development process

To test the performance of our discovery method on simulated data, we consider the designed process model of the part release process, a subprocess of the product development process for Mercedes Benz passenger cars ¹.

We will discover the part release process model from simulated data by using two methods: (i) the discovery method that works under the assumption that log does not contain noise (presented in Chapter 4) and (ii) the discovery method that works under the assumption that log contains noise (presented in Chapter 5). Although these simulated data were not explicitly manipulated for noise (ADONIS simulation component does not generate noisy sequence data), we want to check the robustness of the rule-based model, assuming that incomplete sequences (e.g., when the simulation stops and for some process instances, the process is not yet finished) and imbalance can exist in the simulated logs.

The part release process

As described in Herbst [2000a], the part release process consists on three main reviewing steps: **DMUCheck**, **CheckStandards** and **PMUCheck**. The **CheckStandards** review, concerned with the standards of the drawing, is carried out concurrently with the **DMUCheck** and **PMUCheck**. During the two last mentioned reviewing steps, a part that is to be released is checked against its neighboring parts using digital and physical prototypes. After each of these three reviews, the workers influenced by the results are notified (**informEng+Std**, **informEng+DMU+PMU**). If, for example, the **DMUCheck** fails, the people performing the **CheckStandards** review are informed, because their review become obsolete. If the part submitted to the release process is not in the required state, and also if after the part is finally released, the responsible engineer is notified (**informEngineer**). The process model contains concurrency and duplicate instances for the **informEngineer** and for the **informEng+Std** tasks, as shown in Figure 6.3.

The part release process model is described using the ADONIS representation. When specifying a process model using ADONIS, some graphical blocks are used, as shown in Figure 6.2.

In Figure 6.3, under each task name stands a number, which represents the task identifier. Note that the same task can have more task identifiers (e.g. **informEngineer** is represented by two task identifiers, “52” and “58”). Because our method is not designed to deal with multiple instances of the same task, we will focus on finding a process model whose tasks are identified by these numeric identifiers. In addition to the node types presented in Figure 6.2, the double circle that contains a “V” specifies the existence of a variable that has the value yes/no. For example, after executing the **DMUCheck** task, the *dmuOK* variable will be instantiated with the value “yes” or

¹We want to thank Joachim Herbst for his support to make possible the use of this process model and the generated data.







Graphical representation	Node set	Explanation
	START	Starting node of a process model
	ACT	A task node
	SPLIT	An m of n split (m of n successors may be activated)
	JOIN	Join nodes synchronize the concurrent paths of their corresponding split nodes
	DEC	A decision node. (Exactly 1 of n successors may be activated)
	END	End node of a process model

Figure 6.2: ADONIS node types.

“no”. Another particularity of the part release process model is the existence of two conditions that enable the execution of tasks **MarkPartReleased** or **UnlockPart**. Condition *c1* specifies that if the variables *standardsOK*='yes' and *dmuOK*='yes' and *pmuOK*='yes', then task **MarkPartReleased** is executed. Condition *c2* specifies that if the variables *standardsOK*='no' or *dmuOK*='no' or *pmuOK*='no', then task **UnlockPart** is executed.

The ADONIS simulation component was used to generate a log. We already mentioned that five types of selections are interesting for building process logs, that will be used for discovering the underlying process model. Considering the ADONIS log, we select (i) “nt” (new task) events, (ii) “ts” (task sent) events, (iii) “ta” (task arrives) events, (iv) “st” (start task) and (v) “ft” (finish task) events, that results on five process logs, each containing approximately 900 workflow instances.

Applying the discovery method under the assumption of noise-free process log

Using the resulted five data sets, we employed the α algorithm to build the Petri net process models. The obtained process models are identical for all five selections. The discovered part release process is shown in Figure 6.4. Note the special tasks “b” and “e”, which marks, respectively, the start and the end of processing a case.

Comparing the ADONIS process model from Figure 6.3 with the discovered Petri net model from Figure 6.4, we notice that almost all causal relations are correctly found. Also, the parallelism between tasks “39” and “32” and the decision nodes 'Decision1', 'Decision2' and 'Decision 4' are also correctly found.

However, the ADONIS process model and the Petri net process model are not behaviorally equivalent. The Petri net model cannot result in log traces as, for example,

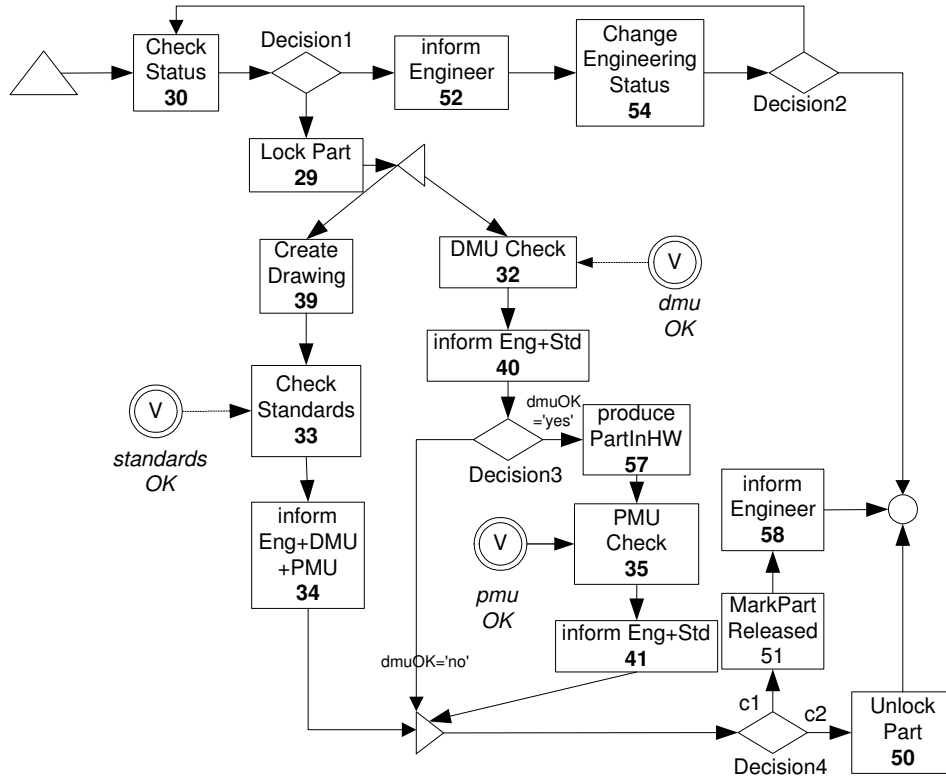


Figure 6.3: The part release process.

"b,30,29,39,33,32,34,40,50,e". If tasks "34" and "40" have already been executed, we have three tokens in three incoming places of task "50"; but task "50" needs four tokens in the incoming places in order to fire, thus this Petri net cannot generate a trace as we considered before.

The problem resides in the way the α algorithm constructs the Petri net. In Figure 6.5 we illustrate this problem for a selection of part release process tasks, in case of the following four causal relations:

1. $40 \rightarrow 57$
2. $40 \rightarrow 50$
3. $41 \rightarrow 50$
4. $41 \rightarrow 51$

First, the α algorithm considers the first and the second causal relations. Because task "40" is directly followed by tasks "50" and "57" and these two tasks do not relate ($50 \neq 57$), the place $p1$ is inserted. Second, the second and the third causal relations are considered, thus task "50" is the direct successor of two tasks, "40" and "41".

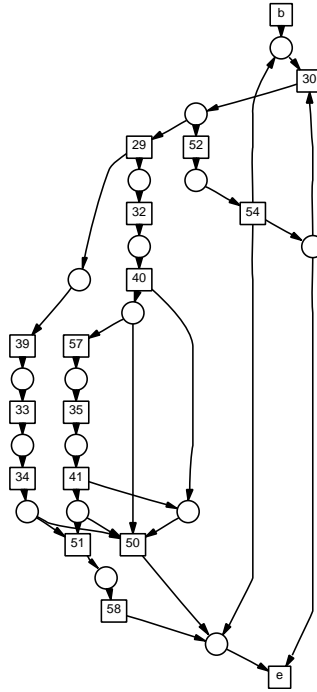


Figure 6.4: The discovered part release process model from simulated data. This process model is build assuming that the process log does not contain noise.

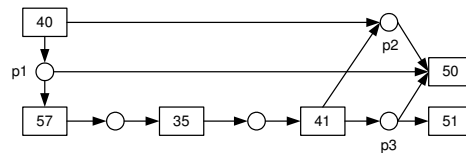


Figure 6.5: The illustration of the α algorithm problem.

Because tasks “40” and “41” do not relate ($40 \# 41$), a new place is added, $p2$. Third, the third and the fourth causal relations are considered, thus task “41” is directly followed by tasks “50” and “51”. Also, tasks “50” and “51” do not relate, thus the place $p3$ will be inserted. In this situation, task “50” is dead, thus it cannot fire: if task “40” is executed, it will place one token in place $p1$ and one token in place $p2$, but there is still needed another token in place $p3$. If we execute tasks 57, 35 and 41, we have two tokens in place $p2$ and one token in place $p3$. But task “50” needs another token in place $p1$ in order to fire, that was already consumed. We need to perform further research in order to improve the modelling capabilities of the α algorithm, to avoid the creation of Petri nets containing dead tasks.

Applying the discovery method under the assumption of noisy process log

Applying the α algorithm is not very useful if we assume that process logs contains noisy sequences. In case of noisy process logs, it is preferable to use first the rule-based models to detect the log-based relations based on the five process logs, and second, to apply the α algorithm to build the Petri net models. We obtain the Petri net process models from Figure 6.6 and 6.7 (see also Figure A.1 from Appendix A).

The discovered Petri nets corresponding to the selection of “nt”, “ts” and “ta” events are similar with respect to the log-based relations, as shown in Figure 6.6 (see also Figure A.1 from Appendix A). The two discovered Petri nets for the selection of “st” and “ft” events slightly differ. For example, in the “nt”, “ts” and “ta” selections, task “32” is directly followed by task “39”, while in case of selecting “st” and “ft”, task “32” is directly followed by task “40”. This shows that selecting different types of events conduce to different process patterns, as we supposed.

However, none of these models can be assimilated with the designed process model presented in Figure 6.3. When comparing the designed model with the discovered models, we focus on detecting the decision points and the parallelism between tasks. Namely, we can notice:

- The three decision points present in the ADONIS process model (Figure 6.3) are correctly detected (i.e. 'Decision1', 'Decision2' and 'Decision4'). However, to model correctly 'Decision3' followed by 'Decision4' it is not possible using the α algorithm. We already discussed this limitation in the previous subsection.
- We could not find the parallelism specified in the designed ADONIS process model, that **CheckStandards** review is done in parallel with **DMUCheck** and **PMUCheck**. However, indirect evidences of parallelism can be found.

Looking closer to the results mentioned above, we have found the choice between tasks “29” (**LockPart**) and “52” (**InformEngineer**) and the choice between tasks “51” (**MarkPartReleased**) and “50” (**UnlockPart**). However, it seems that the path from task “40” (**informEng+Std**) either to task “51” (**MarkPartReleased**) or to task “50” (**UnlockPart**) cannot be depicted. The path from task “40” to task “50” can occur only if either the status variables $standardsOK='no'$ OR $dmuOK='no'$ OR $pmuOK='no'$. Actually, inspecting the five logs, we see that these two tasks occur very seldom next to each other, (e.g. $|40 > 50|=5$ in the “ft” selection). The path from task “40” to task “51” can occur only if the status variables $standardsOK='yes'$ AND $dmuOK='yes'$ AND $pmuOK='yes'$. Thus, it is impossible to have any occurrence “40,51” (because a path from “40” to “51” requires the status variable $dmuOK$ to have the value 'yes', but actually this value is set to 'no' in the designed process model). This fact is also reflected in the process logs, i.e., $|40 > 51|=0$.

The part release process has been designed to involve parallelism. Namely, after task “29” (**LockPart**) is executed, tasks “39” (**CreateDrawing**) and “32” (**DMU-Check**) can be executed, in any order (see Figure 6.3). Applying our discovery method, we find that $29 \rightarrow 32$, but we do not find that $29 \rightarrow 39$. Actually, tasks “29” is directly succeeded by task “39” quite seldom ($|29 > 39|=30$ in the “st” selection, $|29 > 39|=36$ in the “ft” selection and $|29 > 39|=0$ in all other selections).

However, we can find other evidences of parallelism. Confronting the Petri net models from Figure 6.6 with those from Figure 6.7, we find the following alternative

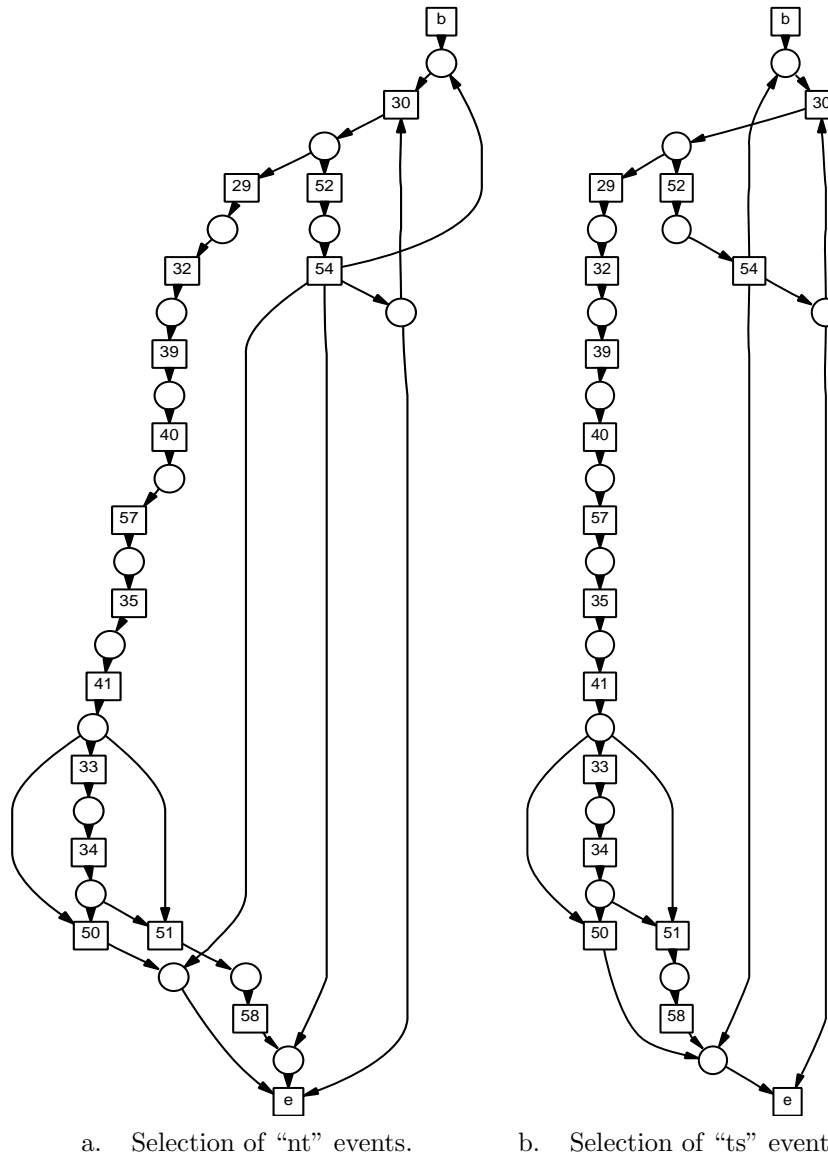


Figure 6.6: The discovered Petri net models for selections of "new task" ("nt") and "task sent" ("ts") events.

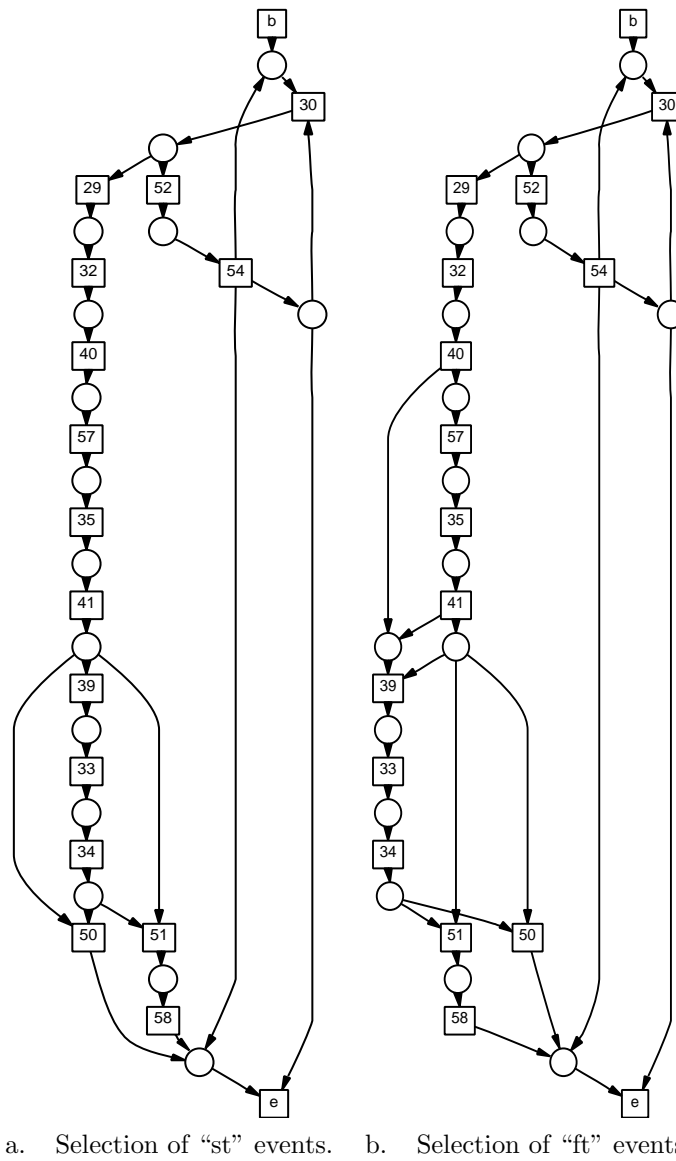


Figure 6.7: The discovered Petri net models for selections of "start task" ("st") and "finish task" ("ft") events.

paths: task “32” is directly followed by task “39” (in Figure 6.6) and by task “40” (in Figure 6.7). Also, task “39” is directly followed by task “40” (in Figure 6.6) and by task “33” (in Figure 6.7). Task “41” is directly followed by task “33” (in Figure 6.6) and by task “39” (in Figure 6.7). We can make the assumption that tasks “39”, “33” and “34” occur in parallel with tasks “32”, “40”, “57”, “35” and “41”. This has to result into non-zero counts of $|X > Y|$, where x are the tasks “39”, “33” and “34” and y are the tasks “32”, “40”, “57”, “35” and “41”. None of these counts are zero, as shown in Table A.1 from Appendix A.

6.1.3 Discussion

Assuming that the process log does not contain noisy data, a Petri net process model can be derived. The ADONIS process model and the Petri net model are comparable with respect to the relations between tasks. However, the discovered Petri net process model will not generate some of the process log sequences because of the modelling limitation of the α algorithm.

When considering the process log as noisy, we observe that:

- different selections of event types conduce to different Petri net models. Although there is not a selection that provides the “best” process model, it seems that the selection of “ft” events results in a process model slightly better than the other selections do (the number of causal relations correctly found is higher for the “ft” selection). One possible explanation is that our discovery method was designed to deal with tasks that are recorded into the process log as soon as they have been completed, and “ft” marks exactly this type of events.
- there is not sufficient support to infer that task “29” is an AND-split task, that is to find that task “29” is directly followed by tasks “39” and “32”. This is due to the high imbalance between the frequency $|29 > 39|$ and the frequency $|29 > 32|$. E.g., in the “ft” selections, $|29 > 39|=36$ and $|29 > 32|=864$ (actually, for all other selections, i.e. “nt”, or “ts”, or “ta”, or “st”, $|29 > 39|=0$). Rather, our discovery method finds that $41 \rightarrow 39$ (in the “st” selection) and $40 \rightarrow 39$ (in the “ft” selection).
- given the counts shown in Table A.1, it seems that tasks “39”, “33”, “34” and tasks “32”, “40”, “57”, “35” and “41” occur in parallel.

Deriving the process model using this simulated log, it seems that performing the **CheckStandards** review in parallel with **DMUCheck** and **PMUCheck** reviews can be seen rather as an exception than a common practice. Actually, the part release process have been previously designed as a sequential process (Herbst [2000b]). Our discovery method seems to provide interesting insights into the process and questions the designed process model. However, a final conclusion can be drawn only by the people that know very good the process.

Another remark about the results of our discovery method can be made in connection with the ADONIS simulation manner. The ADONIS log that we used was not especially generated to be used for testing discovery methods that handle noisy logs. The part release process can truly contain parallel tasks, but the parallel paths

were simulated very imbalanced. Thus, our method is grasping the typical process, rather than the infrequent or exceptional process parts.

We also performed experiments on simulated data using so-called “toy” process models with increasing levels of complexity. The process models have been designed using the same ADONIS modelling tool. These process models were used by the ADONIS simulation component to generate logs. In general, all log-based relations are correctly found, with small exceptions. The original ADONIS process models and the corresponding discovered Petri nets are shown in Appendix A.

6.2 Discovering the process of handling fines

The second application is to test our method on real data resulted from the registration of some enterprise-specific information system.

We choose to find the underlying process of handling fine-collection. The case study is done using data from a Dutch governmental institution responsible for fine-collection ². A *case* (process instance) is a fine that has to be paid. If there are more fines related with the same person, each fine corresponds to an independent case. This process has the particularity that as soon as the fine is paid, the process stops. In total there are 99 distinct activities, denoted by numbers, which can be either manually or automatically executed. We select the fines information for 130136 cases. We construct the process log and we apply to this log our process discovery method that can handle noisy data.

A screen-shot of the discovered process is given in Figure 6.8. Because it is difficult to discuss this complex process model, we focus only on parts of the process. We want to compare the discovered model with the process model resulting from a case study done in the traditional approach, i.e. by interviewing the people involved into the process Veld [2002]. In this study, two sub-processes have been investigated: (i) the COLLECTION sub-process and (ii) the RETURN OF THE UNDELIVERABLE LETTER sub-process.

6.2.1 The COLLECTION sub-process

In Figure 6.9 the process model for the COLLECTION sub-process is presented, as resulting from the case study (Veld [2002]). The process model is designed using workflow modelling blocks. The COLLECTION sub-process starts by receiving from the police the case related documents and then the automated task “initial regulation” (identified by “2”) is executed (e.g., a bank transfer form is sent, specifying the fine amount that must be paid). If after 8 weeks the fine is not paid, a reminder is automatically sent (task “6” - “first reminder”). If the fine is not paid within another 8 weeks, a second last reminder is sent (task “7” - “second reminder”). If after these last 8 weeks the fine is still not paid, a standard verification takes place. This includes the verification of the address done with the help of the Municipal Basic Administration

²The name of the organization is not given for reasons of confidentiality. We want to thank Mr. R. Dorenbos, H.J. de Vries and Hajo Reijers for their valuable support. Also, we want to thank Alexander in 't Veld for his support and efforts relating the data collection and the presented case study.

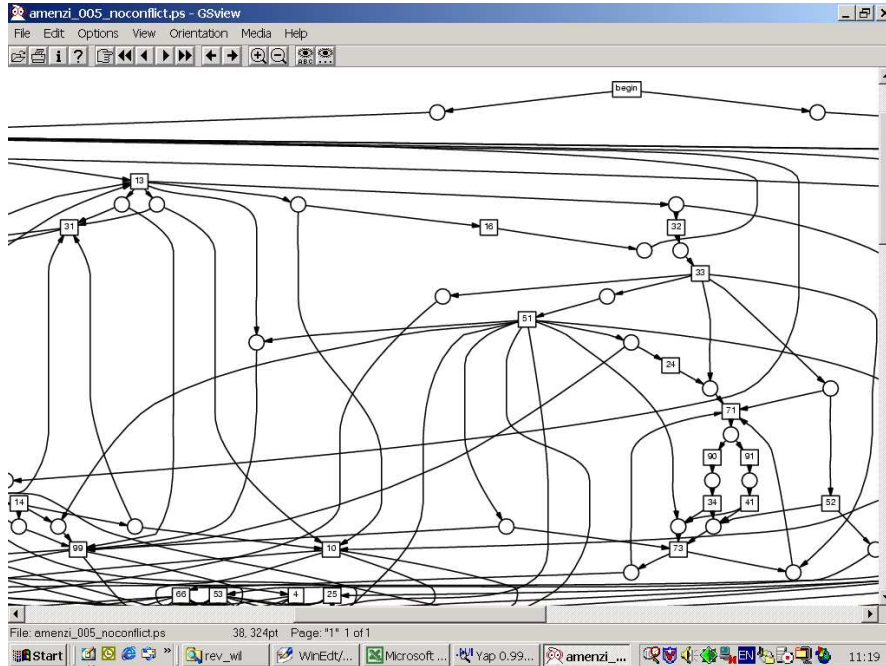


Figure 6.8: Screen-shot for the process.

(MBA) (task “23” - “electronic MBA verification”). The verification is followed by a manual activity, “judge standard verification” (task “13”). Note that after any of tasks “2”, “6”, “7” and “13”, a payment (the task “pay”, represented by an explicit OR-join) can follow, and then the sub-process stops. Task “13” is represented as an explicit OR-split, thus either the payment is made or the sub-process stops.

In Figure 6.10 the discovered Petri net model is shown. Because in the process log it is not recorded a task to mark the completion of cases, we add the task “end” at the end of each trace corresponding to a case. Our method finds that task “2” is directly followed by task “6”, “13” and “end”. Task “6” is directly followed by task “7”. In its turn, task “7” is directly followed by task “23” and “end”, and task “23” is directly followed by task “13”. Subsequently, there is a parallel relation between pairs of tasks (“6”, “13”), (“6”, “end”) and (“2”, “7”) and an exclusive relation between (“23”, “end”) and (“13”, “end”).

Comparing the relations found by our discovery method with the relations from the designed model, we note that in addition to the designed model, our method finds a causal relation between tasks “2” to “13”, that can reveal the existence of an alternate path in practice. Also, in our model, the task “6” (“first reminder”) is directly followed only by task “7” (“second reminder”), and not by the ending task “end” (that would imply the payment, as in the designed model). This can correspond to the fact that only after the seconder reminder people are more willing to pay the fine.

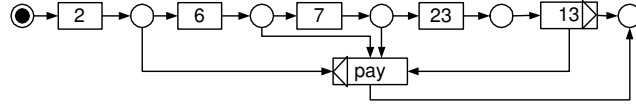


Figure 6.9: The designed workflow net for the COLLECTION sub-process.

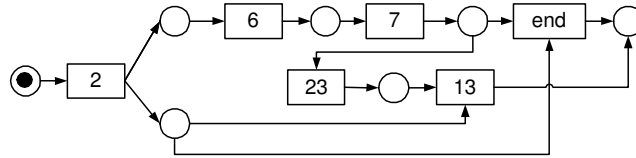


Figure 6.10: The discovered Petri net for the COLLECTION sub-process.

6.2.2 The UNDELIVERABLE LETTER RETURN (ULR) sub-process

In case the person that has to pay the fine cannot be found at the specified address (he/she has moved or deceased), the sanction is called an “undeliverable letter return” (ULR).

There are two reasons that make the comparison between the designed model and the discovered model difficult. First, for the ULR sub-process a workflow model with specific workflow construct is designed in Veld [2002]. Second, in the designed model, tasks were used that do not appear in the process log. In order to make the comparison possible, we focus only on the tasks that appear in the process log and we compare only the causal relations.

The ULR sub-process starts with the task “30” - “undelivered letter return”, that can be directly followed by three tasks: “12”, “13” and “23”. A written verification (“12”) is requested if the sanction is for a company. An electronic MBA verification (“23”) is requested in case of a person. The case can be directly judged by an employee (“13”). This may happen also because a wrong type of verification has been issued. Before the case is leaving the sub-process, it must be anyway judged by an employee, even without verification.

In Figure 6.11 a. and b. are presented the designed model and the discovered model. In both models, task “30” is directly followed by tasks “12”, “13” and “23”. Also in both models, task “13” is directly following tasks “12” and “23”, which is in line with the description made in the previous paragraph. Task “23” is directly followed by task “12” in both models; the explanation can be that when the sanction is for a company, a GBA verification (“23”) instead of a written verification is incorrectly required and only afterwards the written verification is required (“12”).

However, we can note that in case of the designed model, there are also “reversed” direct connections: task “13” is directly followed by tasks “12” and “23” and task “12” is directly followed by task “23”. The explanation can be that such reversed relations can exist, but rather as exceptions than common practice. This reveals that maybe our method is able rather to capture the general process model than the process model containing exceptional paths. We have to conduct more real case

studies in order to ascertain this assumption.

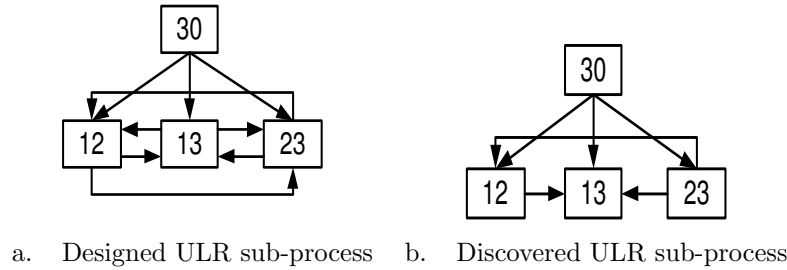


Figure 6.11: The designed and the discovered ULR sub-process in case of selected tasks “30”, “12”, “13” and “23”.

An interesting situation appears when more than one type of relations hold between two tasks. In our case, task “30” causes both tasks “12” and “13”. But task “12” also causes task “13”. α algorithm need to know the relation between the direct successors of task “30”, i.e. “12” and “13”, that can be either an exclusive or a parallel relation. Thus, between tasks “12” and “13” two different relations would hold, (i.e. one causal and one exclusive/parallel relation), which is contradicting the fact that log-based relations are mutually exclusive (see Section 4.3). Actually such a behavior is visible in the log, namely we can see sequences like “30,12”, “30,13” or “30,12,13”. A modelling solution to explain such sequences is given in Figure 6.12. Note that modelling in this way, we will have duplicates for tasks “12” and “13”.

The α algorithm builds the Petri net by considering a unique instance per task and currently cannot model the situation presented before. As we learned from this real world case study, real processes involve tasks with multiple instances, which give us reasons to improve our process discovery method. We leave for future research the improvement of our modelling technique by considering multiple instances of the same task into the Petri net model.

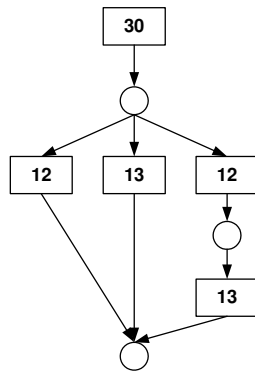


Figure 6.12: A modelling solution for tasks with multiple instances.

6.2.3 Discussion

When discovering both sub-processes, we came to process models comparable with the designed sub-processes. The usefulness of the discovered process model is manifesting in combination with the designed model, i.e. the common parts of these two models can be considered as the “unquestioning” part of the process, while the differences can be used to detect the questionable aspects of the investigated process.

The discovered models have been inspected by the domain experts. They concluded that our discovered models were able to grasp the important aspects of the process. Moreover, the discovered models revealed aspects that are often questioned when discussing the process model. We conclude that process discovery can provide useful insights into the current practice of a process.

6.3 Discovering the treatment process of multi-disciplinary patients

In Chapter 3 we presented the development of logistic homogeneous groups of multi-disciplinary patients. We performed two types of clustering experiments, namely clustering on logistic variables, and clustering based on latent factors extracted from logistic variables. Both types of clustering experiments led to three main clusters, of which two hold clear-cut groups of patients: one labelled “moderately complex” patients, while the other holds “complex” patients. The remaining third cluster contains a small number of cases that cannot be assimilated to one of the two valid clusters.

The goal of this case study is to use our discovery method to investigate the underlying processes of the “moderately complex” and “complex” patients. We want to compare these results with the cluster’s characterizations previously obtained in Chapter 3 (see Table 3.4).

6.3.1 Applying the discovery method to multi-disciplinary patient data

In order to apply the discovery method, we have to construct the process log. We consider as process activities the visits to different specialisms (i.e. surgery, internal medicine etc.), functional investigations and radiology departments³. A case for this process is the *medical case*, i.e. a medical complaint relating a peripheral arterial vascular problem. As we mentioned in Chapter 3, we build with the aid of medical specialists a set of heuristic rules for splitting the patient’s history into separate medical cases. A patient can have more medical cases. The end result was a database with 4395 records as medical cases of the 3603 considered patients. We build two process logs for medical cases from “moderately complex” and “complex” clusters and we applied our process discovery method that can handle noisy data to these logs.

³In this case study we used the following codes for specialisms: CHR - surgery, CRD - cardiology, INT - internal medicine, NRL - neurology, NEUR - neurosurgery, OGH - ophthalmology, LNG - pulmonology, ADI - dialysis, FNKT (FNKC) - functional investigations. RONT (ROEH, RMRI, RKDP, ROZA) stands for activities performed at the radiology department.

In order to simplify the discussion of these models, we considered the most frequent activities, i.e. we selected those events whose frequency divided with the frequency of all events exceed the threshold of 0.01. We show in Figure 6.13 the discovered Petri net model using all medical cases, in Figure 6.14 the discovered Petri net model using only medical cases from the cluster “moderately complex” and in Figure 6.15 the discovered Petri net model using only medical cases from the cluster “complex”.

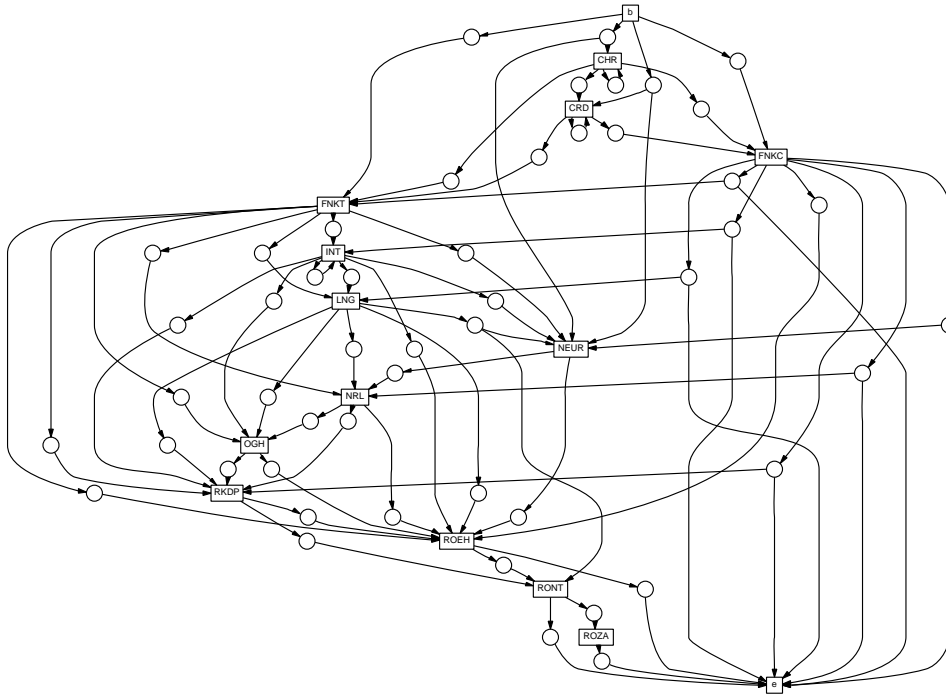


Figure 6.13: The discovered process for all medical cases. The node labels have the following meanings: CHR - surgery, CRD - cardiology, INT - internal medicine, NRL - neurology, NEUR - neurosurgery, OGH - ophthalmology, LNG - pulmonology, ADI - dialysis, FNKT (FNKC) - functional investigations. RONT (ROEH, RMRI, RKDP, ROZA) - radiology.

We observe that in case of “moderately complex” medical cases (Figure 6.14), on every possible path, at most three different specialisms are visited, e.g. “CHR, INT” or “CRD, NEUR, NRL” (the visits for functional investigations and to the radiology departments are not counted as specialisms). Note the existence of a place in case of CHR, CRD, INT and NRL departments, that has the same transition as input and output. Although CHR, CRD, INT and NRL are dead transitions (they need at least two tokens to fire, but in the place with the same transition as input and output it is not possible to be any token) and subsequently these transitions cannot fire, our method indicates a possible self loop. Such loops are very likely to occur, because these specialisms are visited in a repeatedly manner.

In case of “complex” medical cases (Figure 6.15), three or more specialisms are

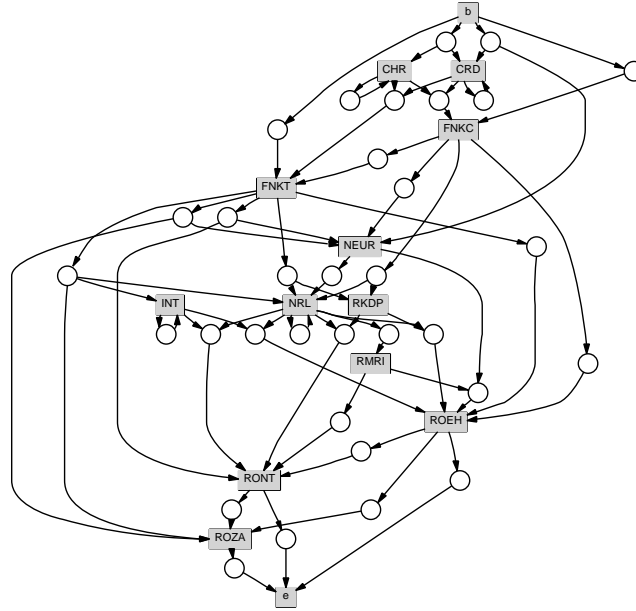


Figure 6.14: The discovered process for cluster “moderately complex” medical cases. The node labels have the following meanings: CHR - surgery, CRD - cardiology, INT - internal medicine, NRL - neurology, NEUR - neurosurgery, FNKT (FNKC) - functional investigations. RONT (ROEH, RMRI, RKDP, ROZA) - radiology.

involved in each possible paths, i.e. “CHR,CRD,INT”, or “ADI,CHR,INT,LNG”, etc. Note that more specialisms are involved in the treatment process of these “complex” medical cases, e.g. LNG, OGH and ADI are added. To emphasize the visited departments that patients from the cluster “moderately complex” (Figure 6.14) have in common with the patients from cluster “complex” (Figure 6.15), we colored in grey the common Petri net nodes.

6.3.2 Discussion

The discovered process models for the “moderately complex” and “complex” clusters confirm the cluster characterization presented in Table 3.4. Namely, patients from the “moderately complex” cluster have visited up to three different specialists, while patients from cluster “complex” have visited more than three different specialists.

In Chapter 3 was mentioned that for a better coordination of patients within hospitals, there is the need to create new multi-disciplinary units, in which different specialties coordinate the treatment of specific groups of patients. According to our results, it seems that for patients with peripheral arterial vascular diseases, two multi-disciplinary units can be created. In case of “moderately complex” cases, a unit consisting in CHR, CRD, INT, NRL and NEUR would suffice. The “complex” cases would additionally need in their treatment specialisms like OGH, LNG and ADI. We note that both clusters share almost the same functional investigation and radiology

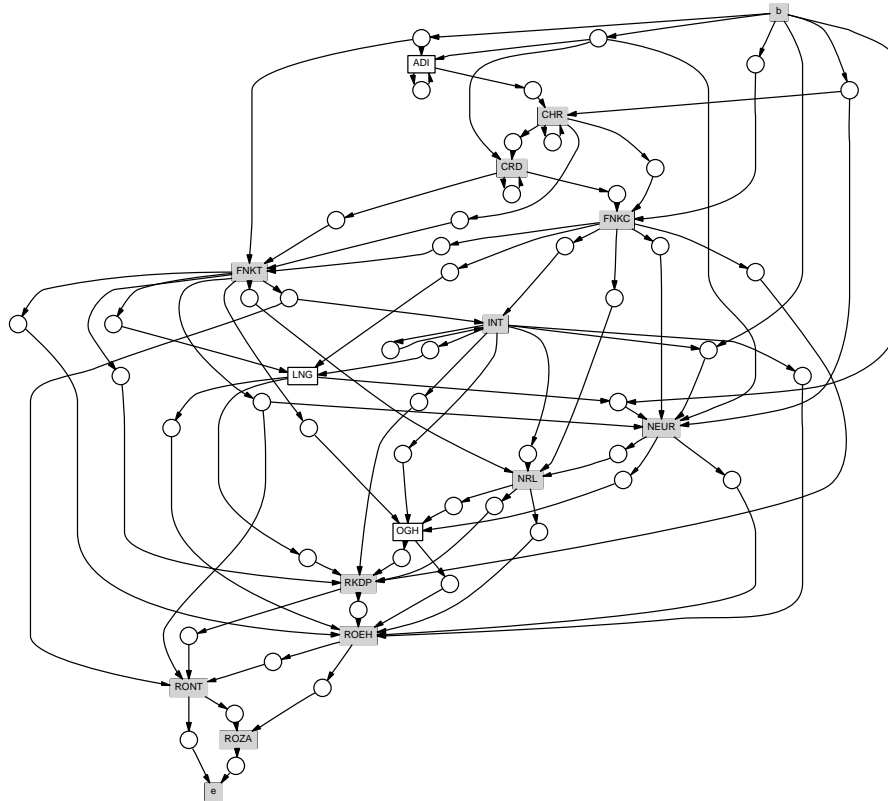


Figure 6.15: The discovered process for the cluster “complex” medical cases. The node labels have the following meanings: CHR - surgery, CRD - cardiology, INT - internal medicine, NRL - neurology, NEUR - neurosurgery, OGH - ophthalmology, LNG - pulmonology, ADI - dialysis, FNKT (FNKC) - functional investigations. RONT (ROEH, RMRI, RKDP, ROZA) - radiology.

departments.

Another requirement specified in Chapter 3 is that adequate selection criteria must exist to select new patients for treatment in a multi-disciplinary unit. Inspecting the predictive rules developed in Section 3.6 (see Table 3.9), we remark Rule #2 for cluster-3 (e.g. cluster “complex”), specifying that if a patient has diagnosis g585 (renal failure), it is likely to be a complex patient. This fact should be consistent with our discovered models. Indeed, if we inspect the discovered model from Figure 6.15, for “complex” medical cases, we note the existence of ADI (dialysis department). Thus, when a new patient comes to be treated in a multi-disciplinary unit, knowing that he/she has renal failure (and consequently, he/she needs to visit the ADI - dialysis- department), it is likely to be a “complex” patients. Assigning the patient to this cluster, it is also possible to know more about the patient’s future route and to estimate what departments are likely to be visited and in what order.

The discovered process models presented in Figure 6.14 and 6.15 leave the im-

pression that both processes contain tasks that occur in parallel. This is likely to be the case. When treating multi-disciplinary patients, many possible combinations of specialisms appear in the log, due to a complex and difficult diagnosis process. Subsequently, it would be very difficult to distinguish between groups of patients, without clustering patients in different groups. This would come to lots of spent resources, even when it is not the case (e.g. for the multi-disciplinary patients that are in the “moderately complex” cluster). Inspecting the discovered process model using all medical cases, shown in Figure 6.13, we can note that a “moderately complex” patient would visit many departments, which will not be necessarily appropriate.

However, none of the three discovered Petri nets are sound Petri nets. This will cause difficulties if one would want to support this hospital process with a workflow management system, based on the discovered Petri nets. The soundness property specifies some critical “correctness” requirements for real processes, e.g. it assures the proper completion of cases, after the completion of an activity, no work is left behind in the workflow (see Section 4.2). Additional research is required to insure the discovery of sound Petri nets from process logs.

6.4 Conclusions

We have shown the application of our discovery method to data from different domains: simulated workflow data, real data resulted from the registration of some enterprise-specific information system and hospital data.

The conclusions of these three applications focus on the following aspects: general conclusions that do not depend on the application, conclusions specific to the application and limitations of the discovery method.

General conclusions. In general, we have been able to discover the investigated processes. Our discovery method is rather able to capture the general process model than the process model containing exceptional paths. More real case studies will be helpful in order to ascertain this assumption.

The discovery method provides reasons to question an existing process design or can reveal new insights into the considered process. The usefulness of the discovered process model is especially manifesting in combination with the designed model.

Conclusions specific to the application. If it is possible to select different event types, for better insights into the process, it is useful to combine these selections. Although there is not a certain selection that provides the “best” process model, it seems that selecting those events that record the case completion, the resulting process models are slightly better than in case of other selections.

In the medical domain, the discovered process models are consistent with the cluster characterization previously made in Chapter 3. Moreover, the discovered Petri nets provide insights into the process of each patient cluster. Our discovered models provide clues about possible direct connections between different departments. This information can be used for constructing multi-disciplinary units.

Limitations. The α algorithm builds the Petri net considering a unique instance

per task. In general, real processes involve tasks with multiple instances, which give us reasons to improve our process discovery method. We leave for future research the improvement of our modelling technique by considering multiple instances of the same task into the Petri net model. An interesting result is the indication of the self-loop in case of CHR, CRD and INT departments, that seems to be visited in a repeatedly manner. However, we have to improve the α algorithm, in order to avoid dead transition results.

As our applications revealed, in real processes more complex situations than we are aware of could be encountered. Therefore, we plan as future work to perform more real-world case studies and consequently adapt and improve our method by considering other factors that may influence the characteristics of the process logs.

In some situations, the discovered process models are not sound Petri nets. Additional research is required to insure the discovery of sound Petri nets from process logs.

Part IV

Conclusions

Chapter 7

Conclusions and suggestions for further research

7.1 Contributions of this thesis

In this thesis we showed that employing machine learning techniques to learn models from data is useful to provide insights into business processes. Given that substantial amount of business process information are recorded, these data can be helpful for gaining a clearer picture about the business process in charge. However, extracting and representing relevant information, and then building useful models is not a trivial process.

Answering to the first research question, e.g. “what data representations can be useful for modelling business processes”, we found useful to represent process information as two sorts of data: *aggregated* data and *sequence* data.

- Aggregated data resulted from transforming raw data into new attributes/variables that measure a certain goal concept, that is not yet explicit in the raw data. In this thesis, we focused on the concept of logistic complexity.
- Sequence data describe the sequencing of activities in a process execution. Sequence data are information recorded in a process log, as the process steps actually have been executed.

This distinction between aggregated and sequence data conducted to modelling from two main perspectives, described in the following two subsections.

7.1.1 The use of aggregated data

We provided the answer at the second and the third research questions, e.g.

2. How can machine learning techniques be used for the clustering of process-related measures?
3. Knowing that relevant clusters can be developed, how can they be used to make predictions?

In Chapter 3 we proposed a methodology that attempts to offer a solution for a better logistic coordination of multi-disciplinary patients with peripheral arterial vascular (PAV) diseases.

In response to the second research question, we showed that patient raw data can be aggregated in six variables, by operationalizing the logistic complexity concept. Subsequently, using clustering technique and principal component analysis, peripheral vascular patients are grouped in two clear-cut clusters. Characterizing the obtained clusters by inducing rule sets, we could share the peripheral vascular patients in “complex” and “moderately complex” patients.

Answering to the third research question, we showed that clustering models are relevant, if predictive models can be built. Using machine learning techniques, we induced predictive models represented as rules, based on some known a-priori patient characteristics. The rules that assign patients to clusters also provide clues about which of the six logistic variables that represent a medical case are relevant or not, and in which interaction they are relevant.

Distinguishing logistically homogeneous groups appears to be important, because every logistic group can require its own optimal control system. The logistic groups that we have found show that searching for logistic homogeneous patient groups makes sense. Although a whole production control system cannot be based on these logistic groups, our proposed approach should be taken as indicative of its potential.

7.1.2 The use of sequence data

During the execution of the process steps, information are recorded in a process log as sequence data. In Chapter 4 we provided the answer at the fourth research question:

4. What kind of processes can be discovered from past process executions?

As a *formal approach*, we developed a discovery algorithm, called the α algorithm, that construct a Petri net model from noise-free process logs that contains sufficient information (i.e., all tasks that potentially directly follow each other in fact directly follow each other in some trace from the log). Performing some initial experiments, the discovered models came to Petri nets having the same structure as the original models. However, in case of a not free-choice Petri net, some causal relations are missed. These results arouse the interest to investigate the limits of the discovery algorithm.

The rediscovery problem investigates whether using the α algorithm it is possible to rediscover the process, i.e., for which class of process models it is possible to accurately construct the model by looking at their logs. It have been shown that it is impossible to rediscover the class of all Petri nets, but the α algorithm can successfully rediscover a large class of relevant Petri nets, under the circumstances of a noise-free and complete log. An interesting characteristic of the α algorithm is that it constructs the “simplest” Petri net generating the behavior exhibited in the log.

The limitations of the α algorithm consists on several problems. First, it cannot deal with not free-choice constructs. It is also known in the Petri net literature that a lot of undecidable problems for general Petri nets are decidable in case of free-choice Petri nets. The second limitation of this algorithm is that short-loops cannot be depicted considering the current version of the completeness notion. Ideas to overcome

the short-loop problem focus on considering a stronger notion of completeness. In Chapter 5 we answered to the fifth research question:

5. It is possible to extract process models from data?

As a *practical approach*, we provided a method that discovers the underlying process from noisy process logs. We generated artificial experimental data by varying the number of task types, noise, execution imbalance and log size. Using these data we aimed to induce models with high accuracy on classifying new data. Two types of models have been induced: a logistic regression model and a rule-based model. The rule-based model for detecting log-based relations significantly outperformed the logistic regression model. We connect this result with the fact that the rule-based model is more flexible than the method based on the logistic regression, which require a fixed threshold value.

We came to a three-step method: the first step employs the rule-based model to detect the causal relations; after the causal relations are found, the second rule-based model detects the exclusive/parallel relations between tasks that share the same cause or the same direct successor. Knowing the causal and exclusive/parallel relations, a Petri net is built that represents the process model. Our rule-sets used in the three-step method have a very high performance in classifying new data, being able to find almost all relations in the presence of parallelism, imbalance and noise. Also, we tested our method on a process log generated by a more complex Petri net than the learning material, resulting in a performance close to that on normal held-out test material.

Using simulating sequence data, we investigated the influence of process log characteristics on our model performance. The causal relations can be predicted more accurately if there is less noise, more balance and more cases. However, causal relations in a structurally complex Petri net can be more difficult to detect. How process log characteristics are influencing the prediction of exclusive/parallel relations is less clear. It appears that noise is affecting exclusive and parallel relations in a similar way as the causal relations, e.g., if the level of noise is increasing, the accuracy of finding the exclusive/parallel relations is decreasing. The current experimental setting confirmed some of our intuitions, e.g. that noise, imbalance and log size are factors that indeed affect the quality of the discovered model. However, in real processes more complex situations than we are aware of could be encountered.

We have shown the application of our discovery method to different data from three domains: simulated workflow data, real data taken from the registration of some enterprize-specific information system and hospital data.

In general, in all three applications, we have been able to discover a process model conforming the reality. The discovery method provides reasons to question an existing process design or can reveal new insights into the considered process. The usefulness of the discovered process model is especially manifesting in combination with the designed model.

As resulted from the case study with simulated workflow data, if it is possible to select different event types, for a better insight into the process, it is useful to combine these selections. Although there is not a certain selection that provides the “best” process model, it seems that selecting those events that record the case completion, the resulting process models are slightly better than in case of other selections.

Applying the discovery technique that can handle noise in case of hospital data, the resulting process models are consistent with the cluster characterization previously made in Chapter 3. Moreover, the discovered Petri nets provide insights into the process of each patient cluster. Although the discovered process models from Chapter 6 contain dead transitions, our discovery method provided indications about situations where self-loop occur.

7.1.3 Combining the two approaches

Our claim is that better insights into processes can be obtained by combining the two perspectives showed in this thesis.

To illustrate this idea, we reload the discussion over the hospital data, where data can be represented both as aggregated and sequence data. Additionally, information about cases (namely, patient's characteristics) are also available.

In Section 3.1 we presented the reasons of getting insights into the treatment process of peripheral arterial vascular (PAV) patients. This is a good example of multi-disciplinary patients that require the involvement of different specialties for their medical treatment. Consequently, this lead to more efforts regarding the coordination of care for these patients.

The problem is to reorganize the care for multi-disciplinary patients in order to increase the care efficiency, that is to eliminate the redundant and overlapping diagnostic procedures. The proposed solution is the creation of new multi-disciplinary units, in which different specialties coordinate the treatment of specific groups of patients. The first component of this solution is to identify salient patients groups in need of multi-disciplinary care. In our particular case of multi-disciplinary patients, we clustered aggregated data and we found that peripheral vascular patients can be shared in two clear-cut clusters, "complex" and "moderately complex" patients.

The second component is to find those relevant specialties that will constitute the ingredients of the multi-disciplinary units. In such multi-disciplinary units, the care for multi-disciplinary patients is not constrained within single units. For identifying the specialties that form the multi-disciplinary units, we built the process models for the treatment of "complex" and "moderately complex" patients, by employing sequence data. We came to the process models by applying the discovery method that can handle noise (see Chapter 5) on the process logs containing the sequence of patient's visits to different specialisms (i.e. surgery, internal medicine etc.), functional investigations and radiology departments. By constructing the process models for "complex" and "moderately complex" patients, we do not focus only on identifying the involved specialties, but we are interested also on the order in which different departments are visited. Moreover, the process models are represented as Petri nets that can be analysed, namely we can check whether the process models are sound, etc.

The third component supposes to have adequate selection criteria to select new patients for treatment in a multi-disciplinary unit. The existence of two different logistic groups reveal the necessity to make a distinction between patients whose treatment processes differ in complexity. The two induced rule-based models, based on known a-priori patient characteristics (age, chronic diagnosis, number of diagnoses, etc.) can be used to assign a new patient to the right cluster, whenever it is a

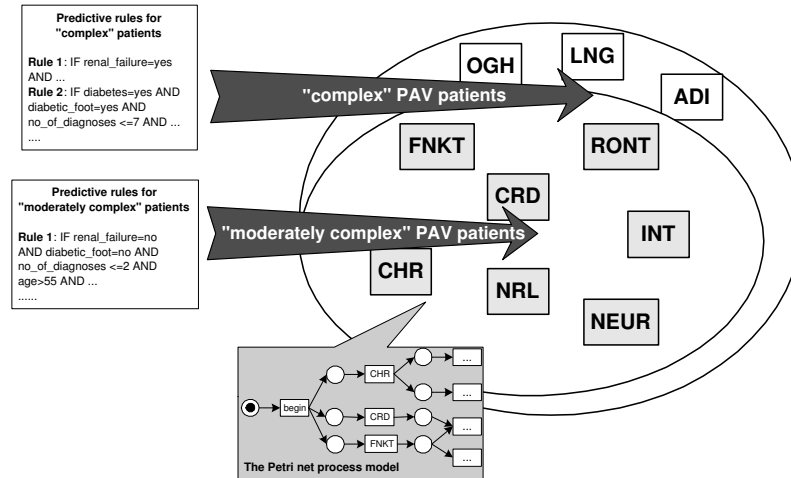


Figure 7.1: Reorganizing the care for multi-disciplinary PAV patients. First, PAV patients are clustered in “complex” and “moderately complex” patients. Second, the process models for the treatment of “complex” and “moderately complex” patients are discovered. Third, two rule-based models are induced to assign new patients to the right cluster. The node labels have the following meanings: CHR - surgery, CRD - cardiology, INT - internal medicine, NRL - neurology, NEUR - neurosurgery, OGH - ophthalmology, LNG - pulmonology, ADI - dialysis, FNKT - functional investigations. RONT - radiology.

“complex” or a “moderately complex” patient.

In Figure 7.1 we provide a visual representation of these three components, which provide a better understanding of the flow of multi-disciplinary PAV patients, and subsequently can be useful in reorganizing the care of these patients.

7.2 Further research

After presenting the main contributions of this thesis, we present some possible directions of future research.

In Chapter 3 we induced predictive models based on some known a-priori patient characteristics. However, the patient characteristics recorded in the hospital information system of our case study were not enough for our purposes. Further research should be invested in finding more a-priori patient characteristics that allow predicting logistic clusters more reliably. We plan to do future research by developing a multi-step model. A-priori knowledge as age, gender, risk factors and relevant secondary diagnosis are known the first time a patient enters the hospital. Based on these information, a first prediction could be made and patients could receive the proper treatment faster. Also, when more information become available through time (as more steps in the process become known), a secondary more precise prediction can be made. Thus, changes in patient groups and treatments could automatically be discovered and relayed back to the logistic management to inspect whether the new

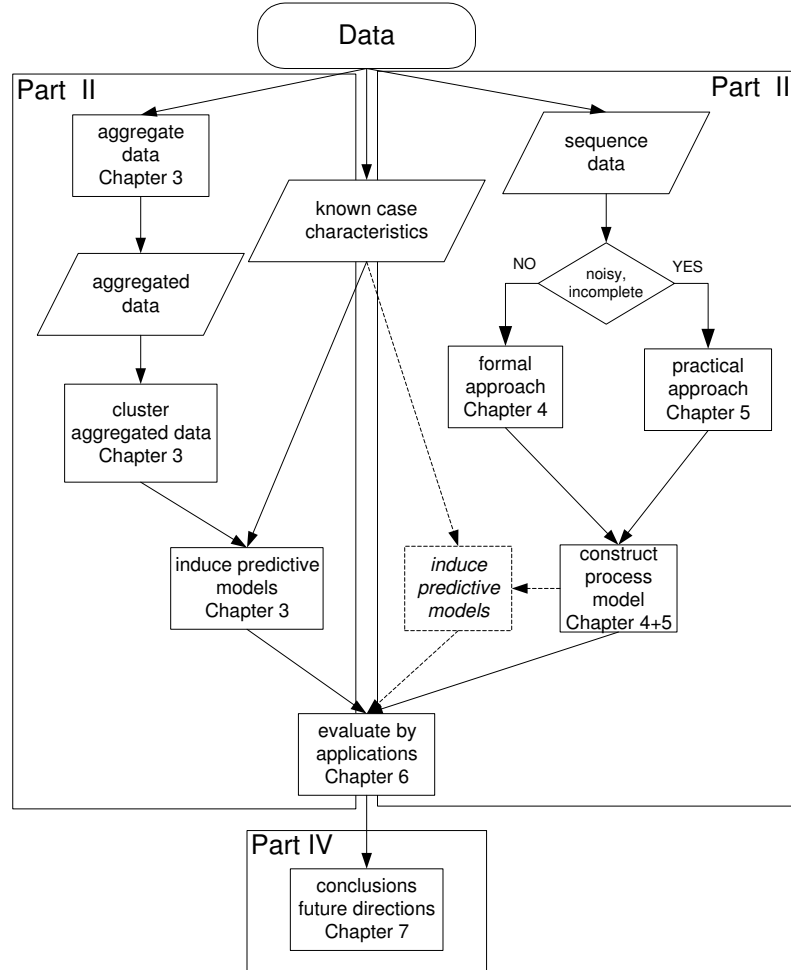


Figure 7.2: The updated research design.

data warrant new changes.

We consider also the possibility of using known case characteristics to predict the future path into the Petri net model. If, for example in the process of handling fines we could know person's characteristics (see Chapter 6), we could induce some predictive models to predict the future route into the process (e.g., a person driving a Ferrari always pay their fines in time). Unfortunately, because of confidentiality reasons, it is difficult to obtain such information. This issue is marked with dotted lines in Figure 7.2, that shows an updated version of the research design presented in Figure 2.1.

Our discovery method is rather able to capture the general process model than the process model containing exceptional paths. More real case studies will be helpful in order to ascertain this assumption. The discovery algorithm presented in Chapter 4 builds the Petri net by considering a unique instance per task. In general, real processes involve tasks with multiple instances, which give us reasons to improve

our process discovery method. We leave for future research the improvement of our modelling technique by considering multiple instances of the same task into the Petri net model. Also, the discovered process models contain dead tasks and are not sound Petri nets. Additional research is required to insure the discovery of sound Petri nets from process logs and to solve the short-loop problem.

As our applications revealed, in real processes more complex situations than we are aware of could be encountered. Therefore, we plan as future work to perform more real-world case studies and consequently adapt and improve our method by considering other factors that may influence the characteristics of the process logs.

Appendix A

Discovering process models from simulated ADONIS logs

Table A.1: The counts of $|x > y|$, where x are the tasks “39”, “33” and “34” and y are the tasks “32”, “40”, “57”, “35” and “41”. This information can support the assumption that x and y are in parallel (see Section 6.1.2).

x	y	$ X > Y $	$ Y > X $
39	32	24	37
39	40	19	277
39	57	16	23
39	35	14	31
39	41	21	489
33	32	7	18
33	40	25	17
33	57	13	13
33	35	16	15
33	41	17	31
34	32	5	5
34	40	16	25
34	57	18	10
34	35	16	12
34	41	20	26

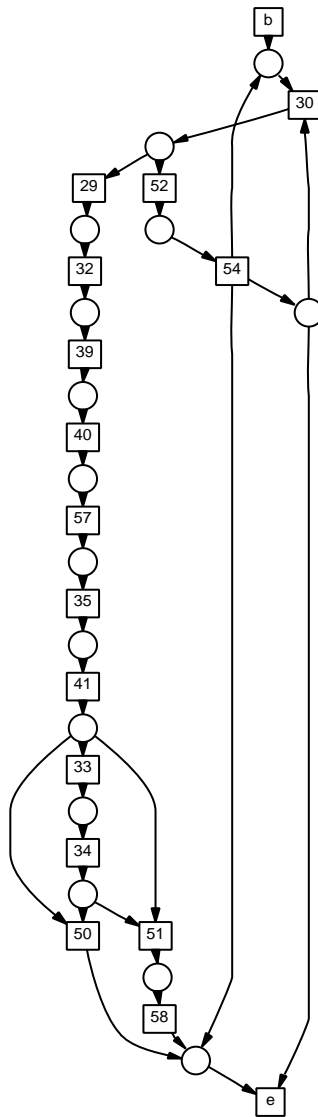


Figure A.1: The discovered Petri net model for the selection of “task arrives” (“ta”) events.

The next eight figures are showing the results of the experiments performed on simulated data, that resulted from “toy” process models. Namely, we used eight ADONIS models, designed with increasing levels of complexity, that involve parallel threads, decision points and cycles. We tried to rediscover these eight designed models from their corresponding logs. An excerpt from the ADONIS log is presented in Chapter 6 (Table 6.1), that was used to discover the model presented in Figure A.2. The causal relations that exist in the designed process model and were missed by the discovery method are marked with dotted arrows.

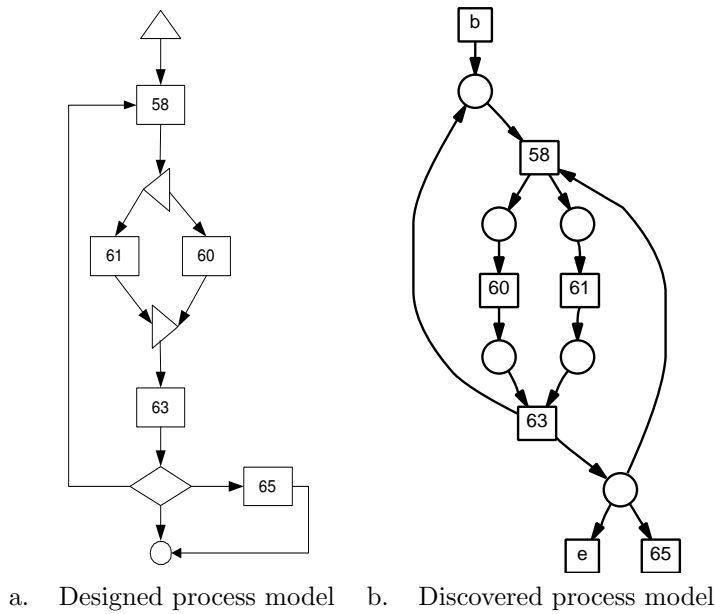
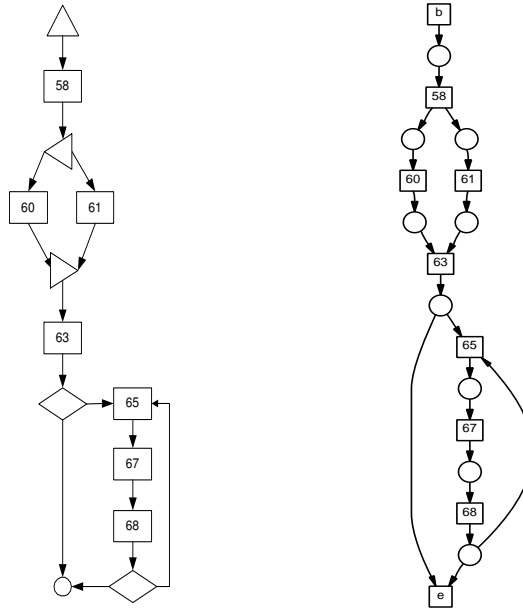
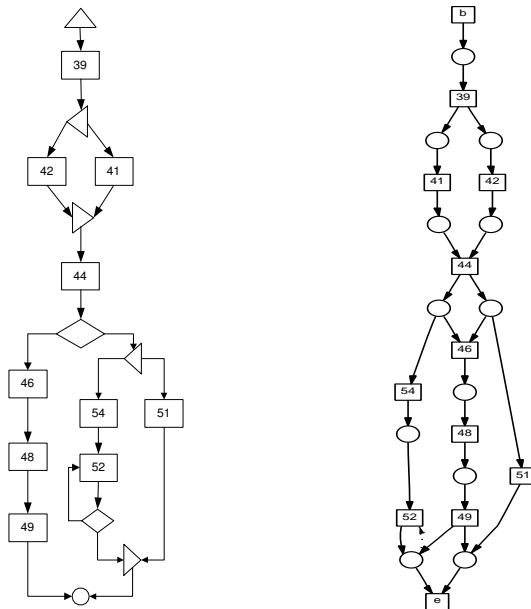


Figure A.2: The designed and the discovered models for example cyc1.



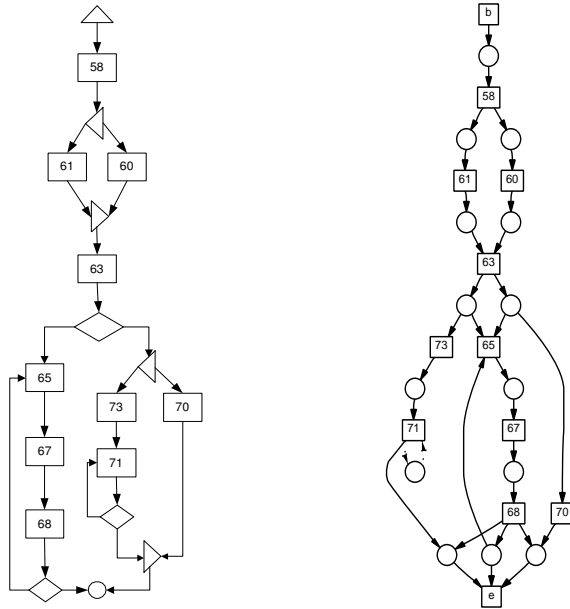
a. Designed process model b. Discovered process model

Figure A.3: The designed and the discovered models for example cyc2.



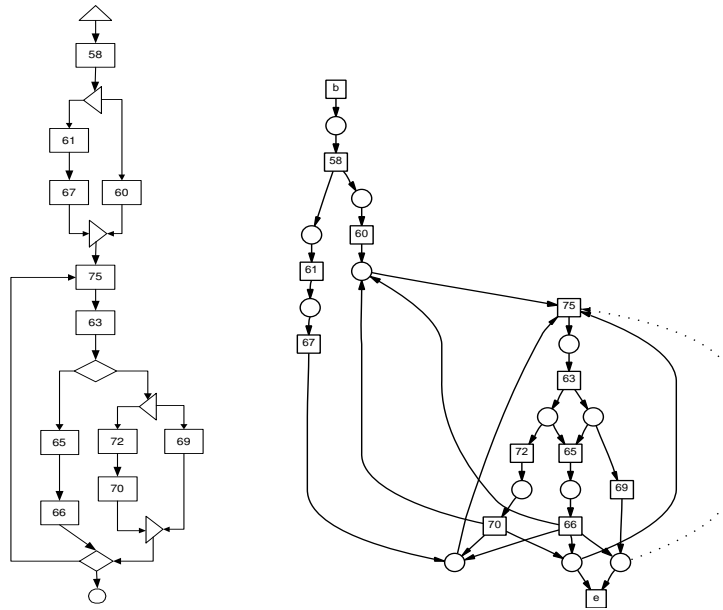
a. Designed process model b. Discovered process model

Figure A.4: The designed and the discovered models for example cyc3.



a. Designed process model b. Discovered process model

Figure A.5: The designed and the discovered models for example cyc4.



a. Designed process model b. Discovered process model

Figure A.6: The designed and the discovered models for example cyc5.

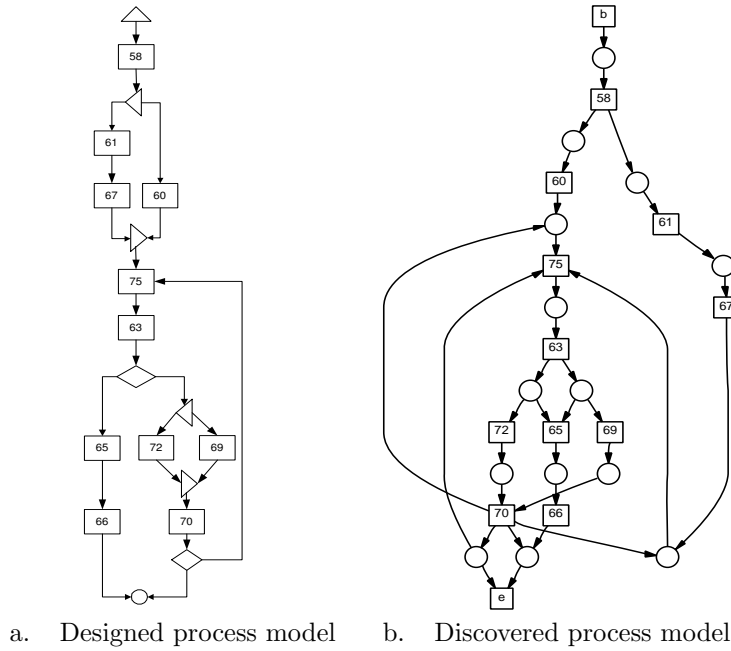


Figure A.7: The designed and the discovered models for example cyc6.

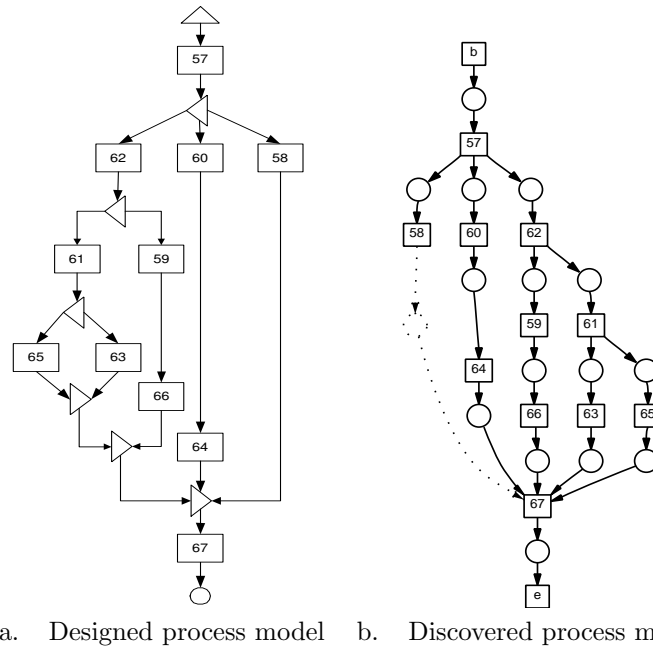


Figure A.8: The designed and the discovered models for example t14.

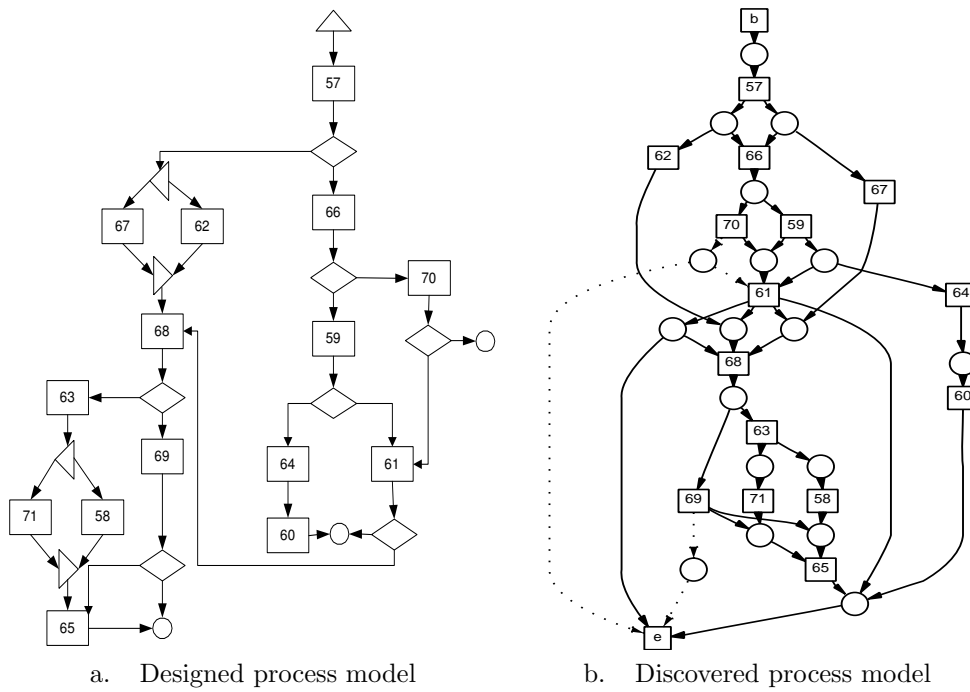


Figure A.9: The designed and the discovered models for example t26.

Appendix B

The rule sets that characterize the logistic clusters

Table B.1: Factor loadings for two latent factors extracted from the original six logistic variables.

	Component	
	Factor-1	Factor-2
<i>C_dif_spm</i>	0.791	-0.027
<i>C_shift</i>	0.908	0.056
<i>N_visit_mc</i>	-0.044	0.890
<i>N_shift_mc</i>	0.056	0.894
<i>M_shift_mth</i>	0.848	0.035
<i>Var_shift_mth</i>	0.829	-0.084

The rule set presented below contains the rules that characterize the clusters that have been developed using all logistic variables. A selection of these rules is presented in Section 3.5.2, in Table 3.4.

Rules for cluster-1

Rule #1 for cluster-1:

if $C_dif_spm \leq 3$
and $C_shift \leq 0.296$
and $N_visit_mc \leq 0.506$
and $M_shift_mth \leq 0.101$
and $Var_shift_mth \leq 0.042$
then cluster-1 (1943, 0.999)

Rule #2 for cluster-1:

if $C_dif_spm \leq 5$
and $C_shift \leq 0.133$
and $N_visit_mc \leq 0.506$
and $M_shift_mth \leq 0.101$
then cluster-1 (1616, 0.999)

Rule #3 for cluster-1:

if $C_dif_spm \leq 3$
and $C_shift \leq 0.353$
and $N_visit_mc \leq 0.506$
and $M_shift_mth \leq 0.101$
and $Var_shift_mth \leq 0.028$
then cluster-1 (1903, 0.999)

Rule #4 for cluster-1:

if $C_dif_spm \leq 4$
and $C_shift \leq 0.449$
and $N_visit_mc \leq 0.688$
and $M_shift_mth \leq 0.074$
and $Var_shift_mth \leq 0.016$
then cluster-1 (1939, 0.996)

Rule #5 for cluster-1:

if $C_dif_spm \leq 3$
and $C_shift \leq 0.296$
and $N_visit_mc \leq 0.506$
and $M_shift_mth \leq 0.224$
and $Var_shift_mth \leq 0.039$
then cluster-1 (1948, 0.996)

Rule #6 for cluster-1:

if $C_dif_spm \leq 4$
and $C_shift \leq 0.25$
and $N_visit_mc \leq 0.506$
and $M_shift_mth \leq 0.101$

then cluster-1 (2089, 0.989)

Rule #7 for cluster-1:

if $M_shift_mth \leq 0.074$
then cluster-1 (2509, 0.89)

Rules for cluster-2:

Rule #1 for cluster-2:

if $N_visit_mc > 0.688$
then cluster-2 (87, 0.978)

Rule #2 for cluster-2:

if $C_dif_spm \leq 6$
and $N_visit_mc > 0.506$
and $M_shift_mth > 0.074$
then cluster-2 (22, 0.917)

Rules for cluster-3:

Rule #1 for cluster-3:

if $C_dif_spm > 3$
and $Var_shift_mth > 0.044$
then cluster-3 (1017, 0.999)

Rule #2 for cluster-3:

if $C_dif_spm > 3$
and $C_shift > 0.232$
and $N_visit_mc \leq 0.688$
and $M_shift_mth > 0.064$
and $Var_shift_mth > 0.03$
then cluster-3 (1216, 0.999)

Rule #3 for cluster-3:

if $C_dif_spm > 3$
and $C_shift > 0.394$
and $Var_shift_mth > 0.01$
then cluster-3 (905, 0.999)

Rule #4 for cluster-3:

if $C_dif_spm > 3$
and $Var_shift_mth > 0.044$
then cluster-3 (1008, 0.999)

Rule #5 for cluster-3:

if $C_shift > 0.211$
and $N_visit_mc \leq 0.506$
and $Var_shift_mth > 0.042$
then cluster-3 (1309, 0.998)

Rule #6 for cluster-3:

if $C_shift > 0.276$
and $N_visit_mc \leq 0.506$
and $M_shift_mth > 0.101$
then cluster-3 (1338, 0.998)

Rule #7 for cluster-3:

if $C_shift > 0.419$
and $N_visit_mc \leq 0.688$
and $M_shift_mth > 0.024$
then cluster-3 (958, 0.997)

Rule #8 for cluster-3:
if $C_shift > 0.583$
and $N_visit_mc \leq 0.688$
then cluster-3 (324, 0.994)

Rule #9 for cluster-3:
if $C_dif_spm > 4$
and $N_visit_mc \leq 0.688$
and $Var_shift_mth > 0.022$
then cluster-3 (873, 0.993)

Rule #10 for cluster-3:
if $C_dif_spm > 5$
and $N_visit_mc \leq 0.688$
then cluster-3 (477, 0.981)

Rule #11 for cluster-3:
if $C_dif_spm > 3$
and $C_shift > 0.304$
and $N_visit_mc \leq 0.688$
then cluster-3 (1303, 0.979)

Rule #12 for cluster-3:
if $M_shift_mth > 0.074$
then cluster-3 (1886, 0.915)

Default : cluster-1

The rule set presented below contains the rules that characterize the clusters that have been developed using two latent factors. A selection of these rules is presented in Section 3.5.3, in Table 3.7.

Rules for cluster-1

Rule #1 for cluster-1

if $C_dif_spm \leq 4$
and $C_shift \leq 0.467$
and $N_visit_mc \leq 0.492$
and $M_shift_mth \leq 0.086$
and $Var_shift_mth \leq 0.03$
then cluster-1 (2165, 1.0)

Rule #2 for cluster-1:

if $C_dif_spm \leq 4$
and $C_shift \leq 0.355$
and $N_visit_mc \leq 0.492$
and $M_shift_mth \leq 0.086$
then cluster-1 (2229, 1.0)

Rule #3 for cluster-1:

if $C_dif_spm \leq 4$
and $C_shift \leq 0.467$
and $N_visit_mc \leq 0.492$
and $M_shift_mth \leq 0.074$
then cluster-1 (2204, 1.0)

Rule #4 for cluster-1:

if $C_dif_spm \leq 6$
and $C_shift \leq 0.225$
and $N_visit_mc \leq 0.604$
and $Var_shift_mth \leq 0.033$
then cluster-1 (1952, 0.999)

Rule #5 for cluster-1:

if $C_dif_spm \leq 5$
and $C_shift \leq 0.205$
and $N_visit_mc \leq 0.604$
and $Var_shift_mth \leq 0.056$
then cluster-1 (1975, 0.999)

Rule #6 for cluster-1:

if $C_dif_spm \leq 3$
and $N_visit_mc \leq 0.476$
and $M_shift_mth \leq 0.136$
and $Var_shift_mth \leq 0.04$
then cluster-1 (2129, 0.999)

Rule #7 for cluster-1:

if $C_dif_spm \leq 4$
and $C_shift \leq 0.379$

and $N_visit_mc \leq 0.476$
and $M_shift_mth \leq 0.136$
and $Var_shift_mth \leq 0.04$
then cluster-1 (2299, 0.999)

Rule #8 for cluster-1:

if $C_dif_spm \leq 5$
and $C_shift \leq 0.333$
and $N_visit_mc \leq 0.604$
and $M_shift_mth \leq 0.104$
and $Var_shift_mth \leq 0.033$
then cluster-1 (2209, 0.998)

Rule #9 for cluster-1:

if $C_shift \leq 0.265$
and $N_visit_mc \leq 0.604$
and $M_shift_mth \leq 0.143$
then cluster-1 (2257, 0.983)

Rule #10 for cluster-1:

if $C_shift \leq 0.333$
and $N_visit_mc \leq 0.604$
and $Var_shift_mth \leq 0.056$
then cluster-1 (2647, 0.951)

Rule #11 for cluster-1:

if $M_shift_mth \leq 0.086$
then cluster-1 (2673, 0.938)

Rule #12 for cluster-1:

if $C_dif_spm > 4$
and $C_shift > 0.32$
and $M_shift_mth \leq 0.086$
and $Var_shift_mth > 0.025$
and $Var_shift_mth \leq 0.027$
then cluster-1 (5, 0.857)

Rules for cluster-2:

Rule #1 for cluster-2:

if $N_visit_mc > 0.838$
then cluster-2 (74, 0.987)

Rule #2 for cluster-2:

if $C_shift > 0.219$
and $N_visit_mc > 0.604$
then cluster-2 (30, 0.969)

Rule #3 for cluster-2:

if $C_shift > 0.333$
and $N_visit_mc > 0.476$
then cluster-2 (24, 0.962)

Rule #4 for cluster-2:

if $N_visit_mc > 0.604$

then cluster-2 (97, 0.859)

Rules for cluster-3:

Rule #1 for cluster-3:

if $C_dif_spm > 4$
and $C_shift > 0.32$
and $Var_shift_mth > 0.027$
then cluster-3 (716, 0.999)

Rule #2 for cluster-3:

if $C_dif_spm > 3$
and $C_shift > 0.239$
and $N_visit_mc \leq 0.604$
and $M_shift_mth > 0.121$
then cluster-3 (938, 0.999)

Rule #3 for cluster-3:

if $C_dif_spm > 5$
and $C_shift > 0.225$
and $M_shift_mth > 0.086$
then cluster-3 (411, 0.998)

Rule #4 for cluster-3:

if $C_dif_spm > 5$
and $Var_shift_mth > 0.033$
then cluster-3 (420, 0.998)

Rule #5 for cluster-3:

if $C_dif_spm > 3$
and $N_visit_mc \leq 0.604$
and $M_shift_mth > 0.121$
then cluster-3 (951, 0.996)

Rule #6 for cluster-3:

if $C_dif_spm > 4$
and $M_shift_mth > 0.086$
and $Var_shift_mth > 0.033$
then cluster-3 (737, 0.996)

Rule #7 for cluster-3:

if $C_shift > 0.15$
and $N_visit_mc \leq 0.604$
and $Var_shift_mth > 0.07$
then cluster-3 (532, 0.996)

Rule #8 for cluster-3:

if $C_shift > 0.3$
and $N_visit_mc \leq 0.604$
and $M_shift_mth > 0.136$
then cluster-3 (978, 0.995)

Rule #9 for cluster-3:

if $C_dif_spm > 6$
and $N_visit_mc \leq 0.604$
and $M_shift_mth > 0.086$

then cluster-3 (210, 0.995)

Rule #10 for cluster-3:

if $C_dif_spm > 5$
and $C_shift > 0.267$
then cluster-3 (427, 0.995)

Rule #11 for cluster-3:

if $C_dif_spm > 5$
and $N_visit_mc \leq 0.492$
and $N_shift_mc > 0.0$
then cluster-3 (433, 0.989)

Rule #12 for cluster-3:

if $C_shift > 0.706$
and $N_visit_mc \leq 0.492$
then cluster-3 (65, 0.985)

Rule #13 for cluster-3:

if $C_dif_spm > 4$
and $N_visit_mc \leq 0.604$
and $M_shift_mth > 0.086$
then cluster-3 (782, 0.982)

Rule #14 for cluster-3:

if $C_dif_spm > 3$
and $C_shift > 0.32$
and $Var_shift_mth > 0.022$
then cluster-3 (1119, 0.961)

Rule #15 for cluster-3:

if $M_shift_mth > 0.086$
then cluster-3 (1722, 0.837)

Rule #16 for cluster-3:

if $C_shift \leq 0.219$
and $N_visit_mc > 0.604$
and $Var_shift_mth > 0.013$
then cluster-3 (6, 0.75)

Default : cluster-1

Appendix C

The rule set for detecting causal relations

The RIPPER.CAUS rule-based model for detecting causal relations. A selection of rules that have a coverage higher than 100 positive instances is presented in Section 5.5.2.

```
IF LM>=0.771 AND LM>=0.913 AND LM>=0.949 AND XY>=0.081 (10797/0)
IF LM>=0.741 AND LM>=0.865 AND YX<=0 AND GM>=0.224
THEN class c (1928/6)
IF LM>=0.741 AND LM>=0.844 AND CM>=0.214 AND CM<=0.438 AND
CM>=0.263 AND GM>=0.038 THEN class c (525/1)
IF LM>=0.741 AND GM>=0.136 AND YX<=0.009 AND CM>=0.267
AND CM<=0.59 THEN class c (337/0)
IF XY>=0.194 AND XY>=0.6 AND CM<=0.827 THEN class c (536/0)
IF XY>=0.204 AND XY>=0.571 AND LM<=0.57 AND CM>=0.092
THEN class c (8/0)
IF LM>=0.702 AND GM>=0.36 AND YX<=0.009 THEN class c (273/0)
IF LM>=0.702 AND LM>=0.812 AND GM>=0.461 AND CM<=0.96
THEN class c (142/0)
IF LM>=0.649 AND LM>=0.757 AND CM>=0.212 AND CM<=0.52 AND
YX<=0.004 AND XY>=0.022 THEN class c (226/27)
IF LM>=0.576 AND CM>=0.255 AND GM>=0.102 AND YX<=0.015 AND
CM<=0.553 AND XY<=0.092 THEN class c (37/1)
IF LM>=0.649 AND GM>=0.166 AND YX<=0.011 AND CM>=0.423 AND
GM<=0.244 AND LM>=0.796 THEN class c (81/3)
IF LM>=0.649 AND GM>=0.268 AND XY<=0.267 THEN class c (56/2)
IF LM>=0.649 AND CM>=0.253 AND GM>=0.04 AND YX<=0 AND
CM<=0.407 AND CM>=0.277 THEN class c (43/4)
IF LM>=0.74 AND XY>=0.217 AND CM>=0.602 AND XY<=0.412
THEN class c (113/0)
IF LM>=0.576 AND LM>=0.753 AND GM<=0.028 AND CM<=0.085 AND
CM>=0.057 THEN class c (37/2)
IF LM>=0.576 AND CM>=0.162 AND LM>=0.757 AND CM<=0.171 AND
GM<=0.047 THEN class c (14/0)
IF LM>=0.576 AND CM>=0.22 AND GM>=0.2 AND XY<=0.298 AND
XY>=0.138 AND YX<=0.035 THEN class c (17/1)
IF GM>=0.111 AND GM>=0.333 AND YX<=0 AND LM<=0.307 THEN class c (11/0)
IF LM>=0.467 AND CM>=0.253 AND GM>=0.019 AND YX<=0.015 AND
CM<=0.485 AND CM>=0.327 AND CM<=0.41 AND GM<=0.045 AND
YX>=0.002 THEN class c (13/1)
IF LM>=0.467 AND CM>=0.221 AND GM>=0.053 AND YX<=0.01 AND
XY<=0.047 AND CM<=0.576 AND GM>=0.077 THEN class c (13/2)
IF LM>=0.576 AND CM>=0.22 AND LM>=0.757 AND GM>=0.127 AND
YX<=0.015 AND CM<=0.553 AND CM>=0.443 THEN class c (9/0)
IF LM>=0.576 AND CM>=0.24 AND LM>=0.757 AND XY>=0.067 AND
CM>=0.638 AND YX<=0.031 AND XY>=0.174 THEN class c (13/0)
```

```
IF LM>=0.576 AND YX<=0 AND CM>=0.054 AND CM<=0.074 AND
LM>=0.702 AND GM<=0.021 THEN class c (20/0)
IF CM>=0.142 AND GM>=0.009 AND YX<=0 AND CM>=0.256 AND
CM<=0.41 AND CM>=0.359 AND XY>=0.016 AND XY<=0.032 AND
CM>=0.382 THEN class c (18/3)
IF XY>=0.363 AND YX>=0.36 AND CM>=-0.041 THEN class c (55/0)
IF CM<=-0.33 AND GM>=-0.002 AND CM<=-0.806 AND CM>=-1.053
AND CM<=-0.964 THEN class c (21/11)
IF GM>=0.049 AND CM>=0.22 AND YX<=0.005 AND CM<=0.527 AND
XY<=0.075 AND GM>=0.111 AND XY<=0.043 THEN class c (11/1)
IF LM>=0.576 AND CM>=0.16 AND YX<=0.008 AND XY>=0.028 AND
LM<=0.649 AND XY<=0.061 AND LM>=0.649 AND GM>=0.049 AND
CM<=0.26 THEN class c (14/0)
IF LM>=0.467 AND CM>=0.243 AND GM>=0.064 AND YX<=0.018 AND
CM<=0.485 AND CM>=0.327 AND XY<=0.086 AND LM>=0.603
THEN class c (9/0)
IF LM>=0.467 AND GM>=0.606 AND CM<=0.802 AND LM>=0.758 THEN class c (5/0)
IF LM>=0.467 AND YX<=0 AND GM>=0.339 AND GM<=0.353 THEN class c (4/1)
IF CM<=-0.316 AND YX<=0.001 AND CM<=-0.803 AND CM>=-0.843
AND CM<=-0.839 THEN class c (9/7)
IF YX<=0 AND XY>=0.009 AND GM>=0.309 AND XY<=0.125 AND
CM<=0.085 THEN class c (8/1)
default n (325905/195)
Train error rate: 0.08% +/- 0.00% (341577 datapoints)
Hypothesis size: 33 rules and 212 conditions
Learning time: 2492.32 sec
```

References

- R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
- D. Angluin and C.H. Smith. Inductive inference: Theory and methods. *Computing surveys*, 15(3):237–269, 1983.
- J.W.M. Bertrand, J.C. Wortmann, and J. Wijngaard. *Production control. A structural and design oriented approach*. Elsevier, Amsterdam, 1990.
- B. Bhanu and K. Krawiec. Coevolutionary construction of features for transformation of representation in machine learning. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO) - Coevolution Workshop*, to appear.
- I. Bruha. Pre- and Post-processing in Machine Learning and Data Mining. In Paliouras, Karkaletsis, and Spyrouopoulos, editors, *Machine Learning and its Applications - ACAI'99*, volume 2049 of *LNAI*, pages 259–266, 2001.
- A. Bryman. *Doing Research in Organizations*. London, Routledge, 1988.
- Casemix. CaseMix Quarterly of the Patient Classification System Europe organization Web Site. <http://www.casemix.org>, 2001.
- P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
- Clementine. Clementine Datamining System Version 6.0.1. User Guide., SPSS Inc., 2000.
- W.W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth Int. Conference of Machine Learning ICML95*, 1995.
- J.E. Cook and A.L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998a.
- J.E. Cook and A.L. Wolf. Event-based detection of concurrency. In *Proceedings of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6)*, pages 35–45, 1998b.

- J.E. Cook and A.L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. *ACM Transactions on Software Engineering and Methodology*, 8(2):147–176, 1999.
- G. de Vries, J.W.M. Bertrand, and J.M.H. Vissers. Design requirements for health care production control systems. *Production planning & control*, 10(6):559–569, 1999.
- G.G. de Vries, J.M.H. Vissers, and G. de Vries. Logistic control system for medical multi-disciplinary patient flows. *Monitoring, evaluating, planning health services, ORAHS'98*, pages 141–151, 1998a.
- G.G. de Vries, J.M.H. Vissers, and G. de Vries. The use of patient classification systems for production control of hospitals. *Casemix Quarterly*, 2:65–70, 1998b.
- J. Desel and J. Esparza. Free choice petri nets. In *Cambridge Tracts in Theoretical Computer Science*, volume 40. Cambridge University Press, Cambridge, UK, 1995.
- T.G. Dietterich. Machine learning. *Annual Review of Computer Science*, 4, Spring 1990.
- D. Dilts, J. Khamalah, and A. Plotkin. Using Clustering Analysis for Medical Resource Decision Making. *Medical Decision Making*, 15(4):333–347, 1995.
- M.H. Dunham. *Data Mining*. Prentice Hall, 2003.
- J. Eder, G.E. Olivotto, and W. Gruber. A data warehouse for workflow logs. In Y. Han, S. Tai, and D. Wikarski, editors, *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, volume 2480 of *LNCS*, pages 1–15, 2002.
- U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*. London, AAAI Press, 1996.
- R.B. Fetter. The new ICD-9-CM Diagnosis-Related Group classification scheme. HCFA Pub. 03167, Health Care Financing Administration, Washington: U.S. Government Printing Office, 1983.
- R.B. Fetter and A. Averill. Ambulatory visit groups: a framework for measuring the productivity in ambulatory care. *Health Services Research*, 19:415–437, 1984.
- D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving business process quality through exception understanding, prediction, and prevention. In P. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. Snodgrass, editors, *Proceedings of 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 159–168. Morgan Kaufmann, 2001.
- J. Herbst. Dealing with concurrency in workflow induction. In U. Baake, R. Zobel, and M. Al-Akaidi, editors, *European Concurrent Engineering Conference*. Society of Computer Simulation (SCS) Europe, 2000a.

- J. Herbst. Inducing workflow models from workflow instances. In *Proceedings of the 6th European Concurrent Engineering Conference*, pages 175–182. Society of Computer Simulation (SCS) Europe, 2000b.
- J. Herbst. A machine learning approach to workflow management. In *Proceedings 11th European Conference on Machine Learning*, volume 1810 of *LNCS*, pages 183–194, 2000c.
- J. Herbst. *Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen*. PhD thesis, Universität Ulm, November 2001.
- J. Herbst and D. Karagiannis. Integrating Machine Learning and Workflow Management to Support Acquisition and Adaptation of Workflow Models. In *Proceedings of the Ninth International Workshop on Database and Expert Systems Applications*, pages 745–752. IEEE, 1998.
- J. Herbst and D. Karagiannis. An Inductive Approach to the Acquisition and Adaptation of Workflow Models. In M. Ibrahim and B. Drabble, editors, *Proceedings of the IJCAI'99 Workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business*, pages 52–57. Stockholm, Sweden, August 1999.
- J. Herbst and D. Karagiannis. Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. *International Journal of Intelligent Systems in Accounting, Finance and Management*, 9:67–92, 2000.
- W.S. Humphrey and P.H. Feiler. Software Process Development and Enactment: Concepts and Definitions. Technical Report SEI-92-TR-4, Software Engineering Institute, Carnegie Mellon University, Pittsburg, PA, 1992.
- S.D. Hunt. *Modern Marketing Theory: Conceptual Foundations of Research in Marketing*. Southwestern Publishing, 1991.
- S. Jablonski and C. Bussler. *Workflow Management. Modelling concepts, Architecture and Implementation*. International Thomson Computer Press, London, UK, 1996.
- S. Junginger, H. Kühn, R. Strobl, and D. Karagiannis. Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation – ADONIS: Konzeption und Anwendungen. *Wirtschaftsinformatik*, 42(3):392–401, 2000.
- N. Kleiner and J. Herbst. A model for bussines process supporting web applications. In *Proceedings of SSGRR 2002*. L'Aquila, Italy, July 29 - August 4 2002.
- I. Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in Medicine*, 23:89–109, 2001.
- N. Lavrač. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16:3–23, 1993.
- G.F. Luger and A. Stubblefield. *Artificial Intelligence*. The Benjamin/Cummings Publishing Company, 1993.

- L. Märuşter, W.M.P. van der Aalst, A.J.M.M. Weijters, A. van den Bosch, and W. Daelemans. Automated discovery of workflow models from hospital data. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 183–190, 2001.
- L. Märuşter, W.M.P. van der Aalst, A.J.M.M. Weijters, A. van den Bosch, and W. Daelemans. Automated discovery of workflow models from hospital data. In C. Dousson, F. Höppner, and R. Quiniou, editors, *Proceedings of the ECAI Workshop on Knowledge Discovery from Temporal and Spatial Data*, pages 32–37, 2002a.
- L. Märuşter, A.J.M.M. Weijters, W.M.P. van der Aalst, and A. van den Bosch. Process mining: Discovering direct successors in process logs. In S. Lange, K. Satoh, and C. H. Smith, editors, *Proceedings of the 5th International Conference on Discovery Science (Discovery Science 2002)*, volume 2534 of *LNCS*, pages 364–373. Springer-Verlag, Berlin, 2002b.
- L. Märuşter, A.J.M.M. Weijters, G.G. de Vries, A. van den Bosch, and W. Daelemans. Logistic-based patient grouping for multi-disciplinary treatment. *Artificial Intelligence in Medicine*, 26:87–107, 2002c.
- L. Märuşter, A.J.M.M. Weijters, W.M.P. van der Aalst, Antal van den Bosch, and W. Daelemans. Discovering process models from empirical data. Submitted to *Data Mining and Knowledge Discovery*, 2003.
- M.K. Maxeiner, K. Küspert, and F. Leymann. Data mining von workflow-protokollen zur teilautomatisierten konstruktion von prozomodellen. In *Proceedings of Datenbanksysteme in Büro, Technik und Wissenschaft*, pages 75–84. Informatik Aktuell Springer, Berlin, Germany, 2001.
- R.S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the First International Symposium on Information Processing*, pages 125–128. Bled, Yugoslavia, 1969.
- S. Miksch. Plan management in the medical domain. *AI Communication*, 12:209–235, 1999.
- T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1995.
- C.U. Moulines. Introduction: Structuralism as a program for modelling theoretical science. *Synthese*, 130:1–11, 2002.
- M. zur Mühlen. Process-driven management information systems combining data warehouses and workflow technology. In B. Gavish, editor, *Proceedings of the International Conference on Electronic Commerce Research (ICECR-4)*, pages 550–566. IEEE Computer Society Press, Los Alamitos, California, 2001a.
- M. zur Mühlen. *Workflow Handbook 2001, Workflow Management Coalition*, chapter Workflow-based Process Controlling-Or: What You Can Measure You Can Control, pages 61–77. Future Strategies, Lighthouse Point, Florida, 2001b.

- M. zur Mühlen and M. Rosemann. Workflow-based process monitoring and controlling - technical and organizational issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
- T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77, pages 541–580, April 1989.
- N.J. Nilsson. <http://robotics.stanford.edu/people/nilsson/mlbook.html>, October 2001.
- C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für instrumentelle Mathematik, Bonn, 1962.
- M. Ploman. Choosing a Patient Classification System to describe the Hospital Product. *Hospital and Health Services Administration*, pages 106–117, May-June 1985.
- J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann, 1993.
- J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1996.
- H.A. Reijers. *Design and Control of Workflow Processes. Business Process Management for Service Industry*. PhD thesis, Eindhoven University of Technology, 2002.
- W. Reisig and G. Rosenberg, editors. *Lectures on Petri nets I. Basic models*, volume 1491 of *LNCS*. Springer-Verlag, Berlin, 1998.
- M. Sayal, F. Casati, M.C. Shan, and U. Dayal. Business process cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.
- I.D.S. Scheer. ARIS Process Performance Manager (ARIS PPM). <http://www.ids-scheer.com>, 2002.
- G. Schimm. Generic Linear Business Process Modeling. In S.W. Liddle, H.C. Mayr, and B. Thalheim, editors, *Proceedings of the ER 2000 Workshop on Conceptual Approaches for E-Business and The World Wide Web and Conceptual Modeling*, volume 1921 of *LNCS*, pages 31–39. Springer-Verlag, Berlin, 2000a.
- G. Schimm. Process Mining. <http://www.processmining.de>, 2000b.
- G. Schimm. Process Mining elektronischer Geschäftsprozesse. In *Proceedings Elektronische Geschäftsprozesse*, 2001a.
- G. Schimm. Process Mining linearer Prozessmodelle - Ein Ansatz zur automatisierten Akquisition von Prozesswissen. In *Proceedings 1. Konferenz Professionelles Wissensmanagement*, 2001b.
- G. Schimm. Process Miner - A Tool for Mining Process Schemes from Event-based Data. In S. Flesca and G. Ianni, editors, *Proceedings of the 8th European Conference on Artificial Intelligence (JELIA)*. Springer-Verlag, Berlin, 2002.
- SPSS. SPSS for Windows, Release 10, SPSS Inc., 2000.

- C.D. Spyropoulos. AI planning and scheduling in the medical hospital environment. *Artificial Intelligence in Medicine*, 20:101–111, 2000.
- Staffware. Staffware process monitor (spm). <http://www.staffware.com>, 2002.
- StatSoft. The Statistics Homepage. <http://www.statsoft.com/>, 2000.
- A. Tarski. Contributions to the theory of models. *Indagationes Mathematicae*, 16: 572–581, 1947.
- F.W. Taylor. *Scientific Management*. Harper Collins, London, 1947.
- W.M.P. van der Aalst. Verification of workflow nets. In G. Balbo P. Azema, editor, *Application and Theory of Petri Nets 1997*, volume 1248 of *LNCS*, pages 407–426, 1997.
- W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
- W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors. *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *LNCS*. Springer-Verlag, Berlin, 2000.
- W.M.P. van der Aalst, B.F. Dongen, J. Herbst, L. Märuster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *IEEE Transactions in Data and Knowledge Engineering*, 2003.
- W.M.P. van der Aalst and B.F. van Dongen. Discovering Workflow Performance Models from Timed Logs. In Y. Han, S. Tai, and D. Wikarski, editors, *International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS 2002)*, volume 2480 of *LNCS*, pages 45–63. Springer-Verlag, Berlin, 2002.
- W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
- W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Märuster. Workflow mining: Which processes can be rediscovered? BETA Working Paper Series WP 74, Eindhoven University of Technology, 2002.
- W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Märuster. Workflow mining: Discovering process models from event logs. *Transactions on Data and Knowledge Engineering*, 2003, to appear.
- M. van Someren. Model Class Selection and Construction: Beyond the Procrustean Approach to Machine Learning Applications. In Paliouras, Karkaletsis, and Spyropoulos, editors, *Machine Learning and its Applications - ACAI'99*, volume 2049 of *LNAI*, pages 259–266, 2001.
- A.J. Veld. WFM, een last of een lust? (Confidential Report), Eindhoven University of Technology, 2002.
- H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing workflow processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.

-
- F.B. Vernadat. *Enterprise Modelling and Integration: Principles and Applications*. Chapman&Hall, London, 1996.
- J. Vissers. *Patient flow based allocation of hospital resources*. PhD thesis, Technical University of Eindhoven, the Netherlands, 1994.
- A.J.M.M. Weijters and W.M.P. van der Aalst. Process Mining: Discovering Workflow Models from Event-Based Data. In B. Kröse, M. de Rijke, G. Schreiber, and M. van Someren, editors, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2001)*, pages 283–290, 2001a.
- A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data. In V. Hoste and G. de Pauw, editors, *Proceedings of the 11th Dutch-Belgian Conference on Machine Learning (Benelearn 2001)*, pages 93–100, 2001b.
- A.J.M.M. Weijters and W.M.P. van der Aalst. Workflow Mining: Discovering Workflow Models from Event-Based Data. In C. Dousson, F. Höppner, and R. Quiniou, editors, *Proceedings of the ECAI Workshop on Knowledge Discovery from Temporal and Spatial Data*, pages 78–84, 2002.
- S.M. Weiss and N. Indurkya. *Predictive data mining*. San Francisco: Morgan Kaufmann, 1998.
- S.M. Weiss and C.A. Kulikowski. *Computer Systems That Learn*. Morgan Kaufmann, 1991.
- I.H. Witten and F. Eibe. *Data Mining*. Morgan Kaufmann Publishers, 2000.

Acknowledgements

There were four years that I had worked on the research that yielded the results presented in this thesis. This time was not only spent to discover processes, but I also had the opportunity to work with and “discover” many interesting people.

First of all, I have to emphasize the contribution of my supervisors, Prof.dr.ir. Hans Wortmann, Prof.dr. Walter Daelemans, dr. Ton Weijters and dr. Antal van den Bosch, who initiated this project, founded by the Organization for Cooperation between Eindhoven and Tilburg universities (SamenwerkingsOrgaan Brabantse Universiteiten - SOBU). Given the cooperative character of this project, I had the opportunity to combine the different expertise of my supervisors in various domains, as machine learning, neural networks, and process and enterprise modelling. I would like to express my thanks to dr. Ton Weijters, my daily supervisor, who regularly guided and advised me in finding “my way” during this research. I want to thank dr. Antal van den Bosch (Tilburg University) and Prof.dr. Walter Daelemans (Tilburg University, Antwerp University), for their valuable ideas, their continuous interest about my research progress and their sharp and quick feedback in all matters. Definitely, I have to thank my first supervisor, Prof.dr. Hans Wortmann, for his high-level view regarding my research, and especially for his support in these very last moments of my project. I am deeply indebted to my supervisors and to Prof.dr. Eric Postma (University of Maastricht), for their guidance, support, prompt feedback and corrections regarding the elaboration of this thesis. I would like to express my thanks to Prof.dr.ir. Wil van der Aalst, for drawing my interest towards process mining and who acted in fact as a supervisor in this particular domain.

I want to thank my parents for everything what they have done for me; even going far from them, they understood and encouraged me in all my decisions. Without their support, this thesis would not have been possible. I want to show my appreciation to my diploma supervisor, Prof.dr. Viorel Negru from the West University of Timișoara, that opened my “appetite” to Artificial Intelligence.

I would like to express my special thanks to Prof.dr. Wim B.G. Liebrand, (currently, director of SURF organization and formerly appointed at Rijksuniversiteit Groningen), who was supporting me in the enterprise to start a Phd in The Netherlands. Thanks to his efforts, it was possible to benefit of a Socrates scholarship, that contributed to start this research project.

I had the big opportunity to have many colleagues, from both departments of Eindhoven and Tilburg: Dutch and Belgians, but also coming from many other countries as France, Russia, China, Austria, Germany, Brasil, Bulgaria, Turkey, Hungary and also Romania. It was a great experience to discover the commonalities and differences

between ourselves. Especially, I want to mention my roommates Teade de Punter, Christine Pelletier, Vladimir Abramov and Florin Tulba, with whom I had very interesting discussions and I could share my daily thoughts. I would like to thank for their warmth and support to Christine Pelletier, Nick Szirbik and Ana Karla Alves de Medeiros, who were not only very good colleagues, but became also very good friends. Many thanks to the secretaries of the I&T department, Ineke Withagen and Ada Rijnberg, for their helpfulness and also for the nice discussions that we had while picking up a cup of coffee. Also, I want to thank the SOBU manager, Therèse van den Heuvel, for her help to get used and enjoy The Netherlands.

Nevertheless, I had the chance to be surrounded by many compatriots (and not only!), which helped me to fill almost at home. I am deeply indebted to my apartment-mate, Cristina Ivănescu that helped me a lot. I will only mention few of them: Bogdana Drăguț, Anca Molnoș, the families Maxim, Presură and Ciobică, Marton Zelina, Andrei Rareș, Simona Vlad and Călin Ciordaș, Mihaela Iftimi-Ilie, Terry Grevenstuk.

Not at least, I want to mention that it would have been very difficult to manage without the spiritual support of Father Silouan and his wife and of a lot of members from the Christian Orthodox Church from Eindhoven.

Thanks!

Summaries

Summary

Business processes (industries, administration, hospitals, etc.) become nowadays more and more complex and it is difficult to have a complete understanding of them. The goal of the thesis is to show that machine learning techniques can be used successfully for understanding a process on the basis of data, by means of clustering process-related measures, induction of predictive models, and process discovery. This goal is achieved by means of two approaches: (i) classify process cases (e.g. patients) into logistic homogeneous groups and induce models that assign a new case to a logistic group and (ii) discover the underlying process. By doing so, the process can be modelled, analysed and improved. Another benefit is that systems can be designed more efficiently to support and control the processes more effectively.

We target on the analysis of two sorts of data, namely aggregated data and sequence data.

Aggregated data result from performing some transformations on raw data, focusing on a specific concept, that is not yet explicit in the raw data. This aggregation is similar to feature construction, as used in the machine learning domain. In this thesis, aggregated data are the variables that result from operationalizing the concept of process complexity. These aggregated data are used to develop *logistic* homogeneous clusters. This means that elements in different clusters will differ from the routing complexity point of view. We show that developing homogeneous clusters for a given process is relevant in connection with the induction of predictive models. Namely, the routing in the process can be predicted using the logistic clusters. We do not aim to provide concrete directives for building control systems, rather our models should be taken as indicatives of their potential.

Sequence data describe the sequence of activities over time in a process execution. They are recorded in a process log, during the execution of the process steps. Due to exceptions, missing or incomplete registration and errors, the data can be noisy. By using sequence data, the goal is to derive a model explaining the events recorded. In situations without noise and sufficient information, we provide a method for building a process model from the process log. Moreover, we discuss the class of models for which it is possible to accurately rediscover the model by looking at the process log. Machine learning techniques are especially useful when discovering a process model from noisy sequence data. Such a model can be further analyzed and eventually improved, but these issues are beyond the scope of this thesis.

Through the applications of our proposed methods on different data (e.g. hospital data, workflow data and administrative governmental data), we have shown that our methods result in useful models and subsequently can be used in practice. We applied our methods on data-sets for which (i) it was possible to aggregate relevant information and (ii) sequence data were available.

Samenvatting

Bedrijfsprocessen (productie, administratie, ziekenhuizen, enz.) worden steeds complexer en het wordt steeds moeilijker om het volledige proces te doorgronden. De doelstelling in dit proefschrift is te laten zien dat automatische leertechnieken met succes kunnen worden ingezet om op grond van historische data meer inzicht te verkrijgen in deze complexe bedrijfsprocessen. Dit doel is bereikt door (i) te laten zien hoe patienten op grond van hun behandeling geclusterd kunnen worden in logistiek homogene groepen en hoe classificatie modellen genduceerd kunnen worden die aangeven tot welke logistieke groep een nieuwe patient behoort (ii) te laten hoe een procesmodel kan worden genduceerd uit een registratie van de uitgevoerde taken. De genduceerde modellen kunnen helpen dit proces te modelleren, te analyseren en mogelijk ook te verbeteren. Proces modellen kunnen op hun beurt gebruikt worden in computer systemen die gebruik maken van een model van het bedrijfsproces dat ze moeten ondersteunen.

Bij het toepassen van de automatische leertechnieken kunnen we twee soorten gegevens onderscheiden en wel afgeleide gegevens (aggregated data) en sequentiele gegevens (sequence data) .

Afgeleide gegevens ontstaan door het uitvoeren van bewerkingen op ruwe gegevens waardoor nieuwe variabelen ontstaan, niet expliciet aanwezig in de oorspronkelijke gegevens. Dit combineren van oorspronkelijke variabelen tot een nieuwe variabele wordt ook wel feature construction genoemd. In dit proefschrift is hiervan gebruik gemaakt om de complexiteit van de behandeling van een patient te operationaliseren in termen van geregistreerde consultaties en behandelingen. De nieuwe variabelen zijn gebruikt patienten te clusteren in logistiek homogene groepen. Dat wil zeggen dat patienten die tot het zelfde cluster behoren wat betreft het doorlopen van het behandelingsproces sterk op elkaar lijken. Voor patienten uit verschillende clusters geldt dit niet. In dit proefschrift laten we zien hoe we hierna regels kunnen induceren die voorspellen tot welk logistiek cluster een nieuwe patient behoort. Het is niet de bedoeling deze regels te gebruiken voor het classificeren van individuele patienten; ze zijn veeleer bedoeld het potentieel voor de gepresenteerde methode aan te tonen.

Sequentiele gegevens zijn niets anders dan een registratie van de volgorde waarin taken worden uitgevoerd. Ze worden opgeslagen in een zogenaamde process log. Inductieve leertechnieken worden in dit geval gebruikt om op grond van deze process-log een bijbehorend proces model te vinden. Door uitzonderingen, onvolledige registratie of andere fouten kan een process-log ruis bevatten. Voor de situatie dat een process-log geen ruis en voldoende voorbeelden bevat, presenteren we een algoritme om op grond van een log een proces model te construeren. Ook laten we zien wat de voorwaarden zijn om te garanderen dat het algoritme het correcte model zal construeren. Wanneer de process-log ruis bevat blijken automatische inductieve leertechnieken goed bruikbaar bij het zoeken naar een proces model. Door analyse en validatie kunnen gevonden modellen mogelijk verbeterd worden, maar deze onderwerpen vallen buiten de doelstelling van dit proefschrift.

Door het toepassen van de in dit proefschrift voorgestelde methoden op gegevens uit verschillende domeinen (zoals workflows, patient registratie, registratie van een administratief proces) laten we de praktische bruikbaarheid van de voorgestelde methoden en de resulterende modellen zien. We hebben ons daarbij moeten beperken tot

gegevens-verzamelingen waarop zinvol 'feature construction' kon worden toegepast of sequentieel waren.

Rezumat

Procesele industriale, administrative, din domeniul medical, etc. devin din ce în ce mai complexe și este tot mai dificil să fie înțelese în totalitate. Țelul acestei teze este de a arăta că tehnicile de tip machine-learning pot fi folosite cu succes pentru înțelegerea unui proces utilizând date. Acest țel este îndeplinit prin intermediul a două abordări: (i) clasificarea cazurilor procesului în cauză (de exemplu, pacienții dintr-un spital) în grupuri omogene din punct de vedere logistic și inducerea de modele care asignează un nou caz la un grup și (ii) descoperirea procesului de bază. În acest mod, un proces poate fi modelat, analizat și îmbunătățit. Un alt beneficiu este posibilitatea de a proiecta sisteme mai eficiente care să fie capabile în mod efectiv să controleze și să susțină procesele reale.

Obiectivul nostru este analiza a două tipuri de date, și anume *datele agregate* și *datele secvențiale*.

Datele agregate rezultă prin transformarea datelor brute, ținând cont de un anumit concept, ce încă nu este reprezentat explicit în datele brute. Această agregare este similară noțiunii de ‘feature construction’ folosită în domeniul machine learning. În această teză, datele agregate sunt variabilele care rezultă prin operaționalizarea conceptului de ‘complexitate de proces’. Aceste date agregate sunt utilizate pentru dezvoltarea de grupuri omogene din punct de vedere *logistic*. Aceasta presupune că elementele aflate în grupuri diferite vor fi deasemenea diferite din punctul de vedere al complexității traseului parcurs în proces. În această teză arătăm că dezvoltarea de grupuri omogene din punct de vedere logistic, în cadrul unui anumit proces dat, este interesantă în conexiune cu inducerea de modele predictive. Adică, folosind aceste grupuri logistice, traseul de parcurs în cadrul procesului poate fi prevăzut. Scopul nostru nu este de a furniza directive concrete pentru construcția de sisteme de control pentru procese; mai degrabă, modelele noastre pot fi considerate ca ilustrative pentru potențialul lor.

Datele secvențiale descriu ordinea în care activitățile au fost executate în timp în cadrul unui proces. Aceste activități sunt înregistrate în cursul execuției într-un fișier jurnal. Din cauza excepțiilor, înregistrărilor lipsă sau a erorilor, datele pot fi afectate de zgomot. Folosind date secvențiale, ținta noastră este să derivăm un model care să explice datele înregistrate. În situația în care datele nu sunt afectate de zgomot și există destulă informație înregistrată în fișierul jurnal, noi furnizăm o metodă pentru construcția unui model de proces. În plus, oferim o discuție asupra clasei de modele pentru care este posibil ca modelul să poată fi descoperit din fișierul jurnal cu acuratețe. Tehnicile de tip machine learning sunt utile în mod special când se pune problema descoperirii unui model de proces din date afectate de zgomot. Un asemenea model poate fi ulterior analizat și eventual îmbunătățit, dar aceste problematice nu fac parte din obiectivele acestei teze.

Aplicând metodele propuse în această teză pe date din domenii diferite (adică date din domeniul medical, date rezultate în urma unor procese de tip workflow și date administrative), arătăm că metodele propuse au ca rezultat modele utile care pot fi aplicate în practică. Aplicarea metodelor propuse s-a făcut pe seturi de date pentru care (i) a fost posibilă agregarea informațiilor și (ii) au fost valabile date secvențiale.

Curriculum vitae

Laura Mărușter was born in Baia Mare, Romania, on July 25th, 1970. After completing in 1989 her pre-university education at the Mathematics-Physics High School “C.D. Loga” of Timișoara, Romania, she started in the same year to study at the West University of Timișoara, Romania, at the Faculty of Mathematics, Department of Computer Science. After graduation, she did her master studies in the same department.

Starting with 1995, she worked as instructor at the West University of Timișoara, Faculty of Sociology and Psychology, teaching disciplines as Informatics and Statistics.

From September 1999, she worked as a trainee research assistant on a project initiated by the Information and Technology group of the Faculty of Technology Management, at the Eindhoven University of Technology and by the Computation Linguistics group of the Faculty of Arts, at the University of Tilburg. The research on applying machine learning techniques on business processes data, carried out in the period September 1999 — August 2003, led to this thesis.

After the defense, which is to take place in August 2003, Laura Mărușter is planning to work as a postdoc researcher at the University of Groningen, Faculty of Management and Organisation, in the Department of Management Information Systems.