Time Series Knowledge Mining

Fabian Mörchen

Dissertation

zur Erlangung des
Doktorgrades der Naturwissenschaften (Dr. rer. nat.),
dem Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg vorgelegt von
Fabian Mörchen aus Dillenburg

Marburg/Lahn 2006

Gutachter:
Prof. Dr. Alfred Ultsch
Prof. Dr. Eyke Hüllermeier

Eingereicht am 7.12.2005
Mündliche Prüfung am 17.2.2006

Abstract

An important goal of knowledge discovery is the search for patterns in data that can help explain the underlying process that generated the data. The patterns are required to be new, useful, and understandable to humans. In this work we present a new method for the understandable description of local temporal relationships in multivariate data, called *Time Series Knowledge Mining* (TSKM). We define the *Time Series Knowledge Representation* (TSKR) as a new language for expressing temporal knowledge. The patterns have a hierarchical structure, each level corresponds to a single temporal concept. On the lowest level, intervals are used to represent duration. Overlapping parts of intervals represent coincidence on the next level. Several such blocks of intervals are connected with a partial order relation on the highest level. Each pattern element consists of a semiotic triple to connect syntactic and semantic information with pragmatics. The patterns are very compact, but offer details for each element on demand. In comparison with related approaches, the TSKR is shown to have advantages in robustness, expressivity, and comprehensibility. Efficient algorithms for the discovery of the patterns are proposed. The search for coincidence as well as partial order can be formulated as variants of the well known frequent itemset problem. One of the best known algorithms for this problem is therefore adapted for our purposes. Human interaction is used during the mining to analyze and validate partial results as early as possible and guide further processing steps. The efficacy of the methods is demonstrated using several data sets. In an application to sports medicine the results were recognized as valid and useful by an expert of the field.

Zusammenfassung

Ein wichtiges Ziel der Knowledge Discovery ist die Entdeckung von Mustern in Daten, die einen Einblick in den generierenden Prozeß geben können. Die Muster sollen neuartig, nützlich und für Menschen verständlich sein. In dieser Arbeit wird eine neue Methodik namens *Time Series Knowledge Mining* (TSKM) vorgestellt, mit der lokale zeitliche Zusammenhänge in multivariaten Daten gefunden und verständlich dargestellt werden können. Für die Repräsentation von zeitlichem Wissen wird die Mustersprache *Time Series Knowledge Representation* (TSKR) definiert. Die Muster sind hierarchisch aufgebaut, jede Ebene entspricht einem anderen zeitlichen Konzept. Auf der untersten Ebene stehen Intervalle, die das Konzept der Dauer repräsentieren. Die Überlappung von mehreren Intervallen stellt auf der nächsten Ebene das Konzept der Gleichzeitigkeit dar. Die Verknüpfung dieser Intervallblöcke mit einer partiellen Ordnungsrelation repräsentiert das Konzept der Abfolge auf der obersten Ebene. Jedes Musterelement besteht aus einem semiotischen Tripel, das syntaktische und semantische Informationen mit einer konkreten Pragmatik verknüpft. Die entstehenden Muster sind durch den hierarchischen Aufbau sehr kompakt, bieten aber bei Bedarf Details zu den einzelnen Elementen. Im Vergleich mit verwandten Ansätzen zeichnet sich die TSKR durch höhere Robustheit, Ausdrucksstärke und Verständlichkeit aus. Effiziente Algorithmen zur Entdeckung der Muster werden vorgeschlagen. Es wird gezeigt, daß sowohl die Suche nach Gleichzeitigkeit, als auch die Suche nach partieller Ordnung als Varianten des bekannten Warenkorb-Problems formuliert werden können. Für die Suche nach häufigen Mustern wurde daher einer der besten bekannten Algorithmen für dieses Problem adaptiert. Der Mensch wird in den Suchprozess eingebunden um Teilergebnisse so früh wie möglich zu analysieren und die weiteren Schritte zu beeinflussen. Die Menge an Mustern wird bewußt kompakt gehalten um den Benutzer nicht mit Informationen zu überfluten. Die Effektivität der Verfahren wird an mehreren Datensätzen demonstriert. Bei einer Anwendung aus dem Bereich der Sportmedizin wurden die Ergebnisse vom Experten als valide und nützlich bezeichnet.

Acknowledgements

I wish to thank my family for the encouragement and welcome distraction during this work, most of all my wife Daniela for her loving support. Many thanks go to my advisor Alfred Ultsch. He guided me in this endeavor with just the right mixture of affirmation and criticism. Numerous ideas of this work have been developed using an example from sports medicine. This was made possible by the close cooperation with Olaf Hoos, whom I thank for repeatedly inspecting the results. I also thank Alan Fern for provision and support of the video data. Finally, I owe a lot to my colleague Katrin Kupas and my friend Adrian Schnobrich for proofreading parts of this work.

Contents

Chapter 1

Introduction

The technological advances of the last decades lead to an omnipresence of computers and networking in many domains of business and science. The fast increasing computing power and the decreasing costs of high volume data storage allowed the collection of enormous amounts of data. This has led to the relatively young field of data mining and knowledge discovery with the goal to make sense of this data and utilize it. Companies seek to learn more about their customers to better target advertisement, predict future product trends, or prevent fraud. Banks and insurances aim to estimate investment risks and to predict the stock market development. Scientists from many domains search for interesting patterns in data observed in experiments or the environment. In medicine, genetic and other data from patients is searched to find the cause of diseases and develop treatments.

Data mining and knowledge discovery can solve many tasks in all these endeavors. Important requirements for knowledge discovery are interpretability, novelty, and usefulness of the results. In many domains the experts want to know why a decision was made, otherwise they are unlikely to trust the advice generated by automated data analysis methods. Novelty of patterns is sought because there is no benefit in deriving information from data, that is already known to the experts. The ultimate goal of knowledge discovery is to foster a deeper understanding of the data in the mind of the analyst and produce results that can be utilized within the application domain.

Many data mining problems involve temporal aspects. The most common form of temporal data is time series where some properties are repeatedly observed generating a series of data items similar in structure. Some examples include stock-trading data, network-traffic, web-server statistics, sensory data, and music archives. Mining this data poses interesting challenges. Whereas the analysis of static data sets often comes down to the question of similarity of data items, with temporal data there are many additional possible relations. Several different data items can be related because they often occur in a particular order or close together in time.

In this thesis we develop a representation of temporal patterns in multivariate time series to support knowledge discovery. The knowledge representation is designed to be easily comprehensible, robust to noise present in the data, and is able to express many different temporal relations. Local temporal phenomena are qualitatively described in a form close to human language. Efficient computer algorithms are presented that find the most dominant patterns in a time series and present a concise summary to the data analyst. This can be used to explain important temporal phenomena of the process generating the data.

Chapter 2 presents our view of data mining and knowledge discovery. The description of temporal data models, operators, and concepts in Chapter 3 provides a common language for the further discussion of temporal data mining. Chapters 4-6 zoom in on the main subject of this study - patterns and rules extracted from time series. Chapter 4 gives a brief overview of temporal data mining. Time series data mining is explained in more detail in Chapter 5.

Existing approaches for temporal rule mining are categorized according to a novel taxonomy in Chapter 6.

The main part of this thesis is the presentation of the Time Series Knowledge Mining (TSKM) method in Chapter 7 and Chapter 8. After a motivation in Chapter 7.1, we define the Time Series Knowledge Representation (TSKR) as a new pattern language for expressing temporal knowledge, in Chapter 7.2. The TSKR is analyzed in comparison to existing approaches in Chapter 7.3. Efficient algorithms for mining the patterns are described in detail in Chapter 8. An experimental evaluation of the TSKM in comparison with competing methods is given in Chapter 9. Many aspects of our work are discussed in Chapter 10, followed by the summary in Chapter 11.

Chapter 2

Fundamentals

This chapter compares several definitions of data mining and knowledge discovery and motivates the emphasis that we place upon understandable results. We briefly describe the most typical data mining tasks, some of which will later occur in variants dealing with temporal data in general or time series in particular.

2.1 Data Mining and Knowledge Discovery

This thesis belongs in the broad context of the still fairly young research area of data mining and knowledge discovery in databases (KDD) that has evolved from a collaboration of mostly the following fields: statistics, machine learning, artificial intelligence, pattern recognition, knowledge acquisition, expert system, data visualization, high-performance computing, databases, information retrieval, multimedia and graphics (Fayyad et al., 1996; Hand et al., 2001; Dunham, 2002). In order to define data mining and knowledge discovery in databases from our point of view, we first quote some other definitions.

An early and often cited definition is given in (Fayyad et al., 1996). The authors define KDD on an abstract level as the *"development of methods and techniques for making sense of data"*. The results are supposed to be *"more compact [...], more abstract [...], or more useful"*. In their view *"KDD refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process"* and *"Data Mining is the application of specific algorithms for extracting patterns from data"*. The patterns are required to be *"valid on new data [...], novel [...], potentially useful [...]"* and *"understandable"*. They emphasize that the *"KDD process is interactive and iterative"*. The basic process is described as the sequence of the steps selection, preprocessing, transformation, data mining, and interpretation/evaluation, but it is said that it *"can involve significant iteration and can contain loops between any two steps"*.

Hand et al. give a definition as follows: *"Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner"* (Hand et al., 2001). They put emphasis on the fact that data sets are usually large, creating algorithmic challenges and distinguishing data mining from classical exploratory data analysis. The relationships found in the data are required to be novel and understandable. For KDD the authors adopt the view of Fayyad et al., but they note that boundaries between knowledge discovery in databases and data mining are not easy to draw because *"to many people data transformation is an intrinsic part of data mining"*. They propose components for a general categorization of data mining algorithms: model or pattern structure, score function, optimization and search method, data management strategy.

Dunham describes data mining from a database perspective and defines it as *"finding hidden information in a database"* (Dunham, 2002). The author notes that knowledge discovery

in databases and data mining *"are often used interchangeably"* but also adopts the view of Fayyad et al. and defines KDD as *"the process of finding useful information and patterns in data"* while data mining is a sub step in the process.

Ultsch defines data mining as the inspection of data with the aim of discovering knowledge. Knowledge discovery is defined as *"the discovery and formal representation of knowledge from data collections"* (Ultsch, 2003a). The terms are thus used somewhat interchangeably, but whenever the latter is used the emphasis is on knowledge. Knowledge is required to be representable in a linguistic form, that is understandable to humans and automatically usable by knowledge-based systems. This can be achieved by using a combination of first order predicate logic and an estimation calculus, e.g., probability or fuzzy theory. Further, knowledge is required to be previously unknown and useful. *"The central theme of data mining and knowledge discovery is the transition from data to knowledge"*. The conversion of sub-symbolic patterns and trends in data to a symbolic form is seen as the most difficult and most critical part of data analysis (Ultsch, 2003a; Ultsch and Korus, 1995).

In accordance with Ultsch, we like to take the terms data mining and knowledge discovery more literally than, e.g., Fayyad et al.. The word *mining* implies a lengthy laboriously process of searching for hidden information in a large amount of data. Since preprocessing often takes up a large amount of the complete analysis time and is also utterly important for getting good results, it seems unfair to define data mining solely as an intermediate step in the process. The term knowledge discovery implies the result to be *knowledge*. In addition to being new, interesting, and useful, we require knowledge to be understandable to humans and automatically interpretable by computers. We also like to drop the suffix *in databases*. During the advent of KDD it was needed to emphasize the focus on large data sets and efficient algorithms compared to exploratory data analysis in statistics. By now, databases have become an omnipresent tool that does not help in defining the concept and the form of storing the data is not crucial for the process. Therefore we want to define the two strongly coupled concepts in a manner somewhat inverse to the traditional way of Fayyad et al., more as a mixture of Ultsch and Dunham:

Data mining The process of finding hidden information or structure in a data collection. This includes extraction, selection, preprocessing, and transformation of features describing different aspects of the data.

Knowledge discovery Data mining with the goal of finding knowledge, i.e., novel, useful, interesting, understandable, and automatically interpretable patterns.

Note, that the definition of the above terms is a rather philosophical question. You might as well think of data mining and knowledge discovery the same concept, but we want to emphasize the strong requirements we pose for a model to be considered knowledge.

2.2 Typical Data Mining Tasks

In the next few sections we briefly describe some typical data mining tasks. The list is not meant to be exhaustive but it does cover many problems encountered in data mining. We will refer to these tasks later in the context of temporal data mining. For more details see (Hand et al., 2001; Dunham, 2002).

Preprocessing

Preprocessing the data to be mined is utterly important for a successful outcome of the analysis. If the data is not cleansed and normalized, there is a high danger of getting spurious and meaningless results. Cleansing includes the removal of outliers, i.e., data objects with extreme

values, replacement of missing values, or the removal of erroneous corresponding data sets. Many data mining algorithms rely on the concept of distance between data sets. Often, a measure of similarity between data sets needs to be chosen. For many similarity measures, e.g., the commonly used Euclidean distance, normalization of the data needs to be considered to avoid undesired emphasis of features with large ranges and variances.

Regression

Regression is the task of explicitly modelling variable dependencies to predict a subset of the variables from others, see (Hastie et al., 2001) for an introduction. The most common case is linear regression, this includes nonlinear transformations of the input variables. A well known form of nonlinear regression is using feed-forward neural networks (Haykin, 1994), recently Support Vector Machines (SVM, Burges (1998)) have become popular. Regression can also be used to replace missing values.

Clustering

Clustering[1] is the process of finding intrinsic groups, called clusters, in a data set. Each cluster should be as homogeneous as possible and distinct from other clusters. A cluster can be defined based on distances or densities. Distance based clusterings have small intra-cluster distances vs. large inter-cluster distances. Density based clusterings have dense clusters separated by relatively sparsely populated regions. The most widely used distance based clustering algorithm is k-Means (MacQueen, 1965), a well known density based clustering algorithm is DBSCAN (Ester et al., 1996).

Classification

Classification is the task of assigning class labels to a data set according to a model learned from training data where the classes are known. It can be viewed as a special case of regression with a discrete prediction variable. In the context of knowledge discovery understandable classifiers are preferred over. The well known feed-forward neural nets can learn class assignments, but the models are built using a large amount of numeric weights that cannot be easily interpreted. Classifiers the represent the model in form of rules or trees with human readable conditions on the feature values (see Witten and Frank (1999) for a summary of common approaches) are better understandable. The most widely used examples are decision trees (Quinlan, 1993).

Association Rules

Association rules (Agrawal et al., 1993b) were originally developed for the analysis of shopping baskets. A set of products purchased by a customer is represented by a set of discrete items, called itemset. Association rules describe the co-occurrence of items in these itemsets. Two classical examples of products that are likely to be bought together are diapers and beer or chips and salsa. The best known algorithm is Apriori (Agrawal and Srikant, 1994). It is based on the observation, that an itemset can only be frequent if all subsets are frequent as well. Using this property, larger frequent itemsets are built from the bottom up merging smaller itemsets. The concept of association rules was generalized to many different application areas and other data types. In order to reduce the number of itemsets found, the search is can be restricted to maximal or closed itemsets (e.g. Zaki and Hsiao (2002)). Maximal itemsets are not a subset of another frequent itemset. For closed itemsets there is no proper superset with the same frequency.

[1] Sometimes clustering is called unsupervised classification.

Retrieval by Content

When retrieving data sets by content the user provides an example or a template for a data set and approximate matches from a large database are listed. Since there is often a large amount of data, the brute force method of calculating the distances of the query to all data sets is often prohibitively slow. Database indexes, e.g., tree structures, are commonly used to speed up this search (Weber et al., 1998). Another useful tool is a fast lower bounding distance. The exact and more expensive distance only needs to be calculated if the lower bounding distance is below a threshold. This way many data sets can be excluded from the result by only calculating the cheap approximate distance.

Chapter 3

Representations of Temporal Data

In this chapter we define a common language for the description of data mining methods that evolved around temporal data. Temporal data models are described in Chapter 3.1. They provide different ways of representing observations involving time. Temporal operators (Chapter 3.2) are used to combine elements of temporal data models to express temporal patterns. We finally introduce the notion of temporal concepts in Chapter 3.3 to provide a unifying view of temporal relations based on the commonly used temporal operators.

3.1 Temporal Data Models

In this chapter we will introduce models for temporal data. All term definitions will be *emphasized*.

Usually temporal data is represented based on observations at discrete points in time. Often, time is uniformly sampled. If not, there is usually some lower bound for the granularity of time. We refer to this finest level of temporal detail as *time points*. Even if no explicit time values but only an ordering of the data is given, this can be mapped to integer values representing time points. A *time series* is a set of unique time points with values or objects assigned to each time point. A *time sequence* is a multi-set of time points with assigned values or objects, i.e., it can include duplicate time points. A pair of time points defines a *time interval*. An *interval series* is a sequence of non-overlapping time intervals. In a *contiguous interval series* no gaps between two consecutive intervals are allowed. An *interval sequence* can include overlapping and even equal time intervals.

A *numeric time series* is a time series with numerical values for each point. This is the data model commonly used in statistics, e.g., for the prediction of stock prices. A *symbolic time series* is a time series with nominal values for each point. This is the data model commonly used in bioinformatics, e.g., for the analysis of gene data and can be obtained from numeric time series by discretization (see Chapter 5.4.3). A *numeric interval series* is an interval series with numerical values for each interval, a *symbolic interval series* has nominal values, respectively. These data models are commonly obtained from univariate numeric time series by segmentation (see Chapter 5.8) and feature extraction (see Chapter 5.2). A symbolic time series can be obtained from a symbolic interval series using, e.g., only the start points of each interval and ignoring the durations.

A *numeric time sequence* is a multi-set of time points with numerical values for each point. A *symbolic time sequence* is a multi-set of time points with nominal values for each point. This data model is used, e.g., in (Mannila and Toivonen, 1996) for telecommunication events. A *numeric interval sequences* provides numerical values for each interval, a *symbolic interval sequences* has nominal or ordinal values.

In all symbolic data models, the symbols can optionally be accompanied by more attributes, describing, e.g., the length of an interval (Last et al., 2001) or the local slope (Höppner, 2002c). The three mentioned series can be *univariate*, i.e., consisting of a single series or *multivariate* when several series cover the same time range. Further, there can be sets of all the above mentioned models, e.g., sets of numeric series (Dugarjapov and Lausen, 2002) or sets of symbolic series in multiple sequence alignment of genetic sequences.

An *itemset sequence* is a time sequence with a subset of a set of symbols per time point. This data model is used in inter-transaction association rule mining (Agrawal and Srikant, 1995). A symbolic sequence is a degenerate form of an itemset sequence.

Another temporal data model mentioned in (Roddick and Spiliopoulou, 2002) is mining results, e.g., a set of discovered association rules that can change over time. In (Morik, 2000) temporal facts are defined as logical predicates involving start and end points of an interval as arguments, in (Padmanabhan and Tuzhilin, 1996) temporal facts are defined for time points.

Literally, the term series and sequence both refer to ordered entities. We chose to define the terms as a set and a multi-set of time points, respectively, adhering to the most typical usage of these terms in the literature. The term time series is widely used in statistics for a set of numeric measurements taken at unique time points. The term sequence is widely used for mining sequential patterns in itemsets, the transaction times of which can include duplicates.

3.2 Temporal Operators

3.2.1 Time Point Operators

The following temporal operators can be used with time series and time sequences. For two points in time there are three basic binary operators: *before*, *equals*, and *after*. Both *before* and *after* can be accompanied by a threshold, e.g., after at most (least) k time units. This corresponds to a complete ordering of the time stamps and is used in many temporal data mining algorithms. Sometimes an operator is used to specify that two or more time stamps lie within a time interval (Mannila and Toivonen, 1996; Das et al., 1998; Weiss, 1999) expressing closeness in time without a particular order.

Bettini et al. define an operator called *temporal constraint with granularity*, that can express the operators *before*, *after*, and *equals* w.r.t. a granularity of time (Bettini et al., 1998). They argue, that one day is not equivalent to 24h, because the latter could cover parts of two days, an important distinction in some applications. Also, finer granularities do not necessarily have to correspond to larger ones: an hour is always part of a day, but not every hour is part of a business day. Thus, the result of the operator can be undefined. A fuzzy extension for temporal reasoning (see Chapter 4.7) has been proposed in (DuBois and Prade, 1989) expressing relations like *much before* or *closely after*.

Periodical patterns (Han et al., 1998) are not commonly expressed with operators, but one could say that a symbol *repeats* approximately every k time points.

3.2.2 Time Interval Operators

Time interval operators can only be used with interval series and sequences.

Allen's Interval Operators

For the purpose of temporal reasoning Allen formalized temporal logic on intervals by specifying 13 interval relations (Allen, 1983) and showing their completeness. Any two intervals are related by exactly one of the relations. The operators are: *before*, *meets*, *overlaps*, *starts*, *during*, *finishes*,

the corresponding inverses *after*, *met by*, *overlapped by*, *started by*, *contains*, *finished by*, and *equals* (see Figure 3.1).

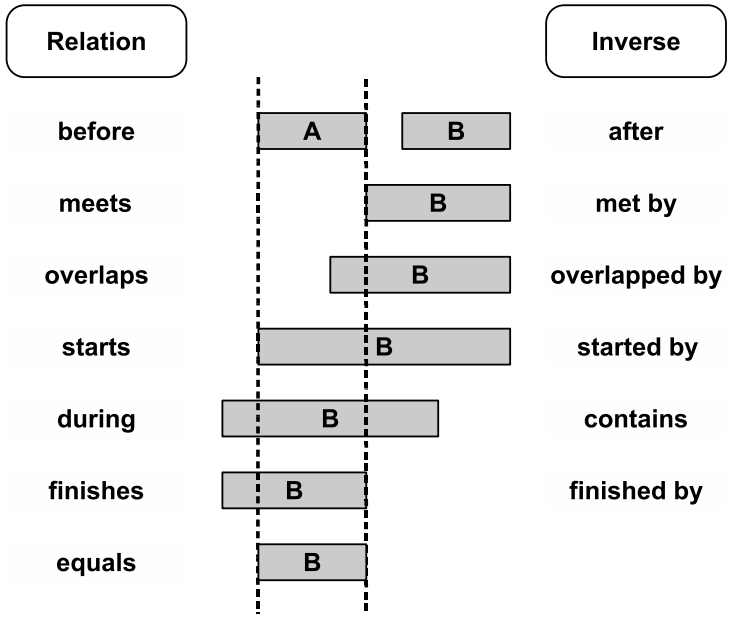


Figure 3.1: Examples of the 13 interval relations between the intervals A and B according to (Allen, 1983). Each relation and the corresponding inverse are shown in one row, e.g., the first rows depicts A *before* B or the alternative formulation B *after* A. The *equals* relation is its own inverse.

The relations are commonly used beyond temporal reasoning, e.g., for the formulation of temporal rules (Kam and Fu, 2000; Cohen, 2001; Höppner, 2002c). Others allow all relations but restrict the composition of several relations (Kam and Fu, 2000; Cohen, 2001) or the maximum duration of a pattern (Höppner, 2002c) to make the search space for patterns smaller.

The strict use of Allen's interval relations has been recognized as problematic. Badaloni and Giacomin propose a fuzzy extension of Allen's interval relations by adding a preference degree to each possible relation (Badaloni and Giacomin, 2000). They discuss the implications on reasoning tasks, but they do not describe how to calculate the membership values for the relations. For other approaches to fuzzy time interval relations see (Calin and Galea, 2001; Nagypal and Motik, 2003; Ohlbach, 2004). Snoek and Worring further object, that two intervals will always have a relation, no matter how far they are apart (Snoek and Worring, 2004). They use a so-called fuzzy version of Allen's relations originally proposed for document processing (Aiello et al., 2002). Using a threshold, interval endpoints are considered equal if they are within a small distance. This way the strict meets, starts, finishes, and equals relations apply more often, while before, overlaps, and during occur less often. For intervals very far apart (determined by a second threshold) they introduce the *no relation* operator. This modified version is called TIME (Time Interval Multimedia Event) relations. There are no fuzzy membership functions, however, only one relations holds for any two given intervals.

Freksa's Semi-Interval Operators

Freksa generalized the interval relations by using semi-intervals (Freksa, 1992) with the following 11 operators: *older*, *younger*, *head to head*, *survives*, *survived by*, *tail to tail*, *precedes*, *succeeds*, *contemporary*, *born before death*, and *died before birth*. Combining the previous leads to six derived operators: *older & survived by*, *younger & survives*, *older contemporary*, *surviving con-*

temporary, *survived by contemporary*, and *younger contemporary*. The generalization was done motivated by the observation that *"in no case, more than two relations between beginnings and endings of events must be known for uniquely identifying the relation between the corresponding events"*. The semi-interval representation has the advantage that coarse or incomplete knowledge is represented by simpler formulas instead of long disjunctions. A concept of similarity of the relations is described. Freksa's semi-interval relations were used to mine temporal association rules in (Rainsford and Roddick, 1999).

Roddick's Midpoint Interval Operators

Roddick and Mooney combined Allen's interval relations with the five point-interval relations of (Vilain, 1982) considering the relative positions of the interval midpoints (Roddick and Mooney, 2005). A total of 49 relations is obtained, e.g., nine different versions of *overlaps*. Some of the different overlaps relations can be interpreted as small, medium, or largely overlapping intervals. The motivation is handling data with coarse time stamps and data from streams with arbitrary local order. The authors also describe the relation between the models of Allen and Freksa and the respective extensions to midpoints and/or intervals of equal durations.

Other Interval Operators

Villafane et al. use an interval operator called *containment*, that is equivalent to the disjunction of Allen's *equals*, *starts*, *during*, and *ends* (Villafane et al., 1999).

Ultsch defines the Unification-based Temporal Grammar (UTG) that contains an approximate version of Allen's *equals* operator called *more or less simultaneous* (Ultsch, 1996b; Guimarães and Ultsch, 1997; Ultsch, 2004). The start and end points of the intervals are not required to be exactly equal, they only need to be within a small time interval. A further generalization, called *coincides*, was proposed in (Mörchen and Ultsch, 2004) by dropping the constraints on the boundary points, only requiring some overlap between the intervals. This is equivalent to the disjunction of Allen's *overlaps*, *starts*, *during*, *finishes*, the four corresponding inverses, and *equals*.

3.2.3 Temporal Logic Operators

Temporal logic operators apply to data given as temporal facts. They evaluate the truth of facts at time points or during time intervals.

In (Bacchus and Kabanza, 2000) first order Linear Temporal Logic (LTL) is used for planning tasks. LTL is an extension of First Order Logic (FOL) used to specify temporal facts about predicates in a sequence of worlds. The main temporal operators are *until* and *next*, additionally the derived operators *always* and *eventually* are used. Padmanabhan and Tuzhilin use an extension called First Order Temporal Logic (FOTL) with the core operators *since*, *until*, *next*, and *previous* (Padmanabhan and Tuzhilin, 1996). Additionally they derive the operators *always*, *sometimes*, *before*, *after*, and *while* and introduce the bounded versions $until_k$ and $since_k$. The operators relate facts instantiated at time points. Cotofrei and Stoffel use a first order language with the temporal base operators *sometimes*, *always*, *next*, and *until* and some derived operators to express temporal classification rules (Cotofrei and Stoffel, 2002c).

In (Rodriguez et al., 2000) three interval predicates based on a visual rule language (Alonso and Rodriguez, 1999) are used: *always*, *sometime*, and *true-percentage*. The predicates explicitly include start and end points of an interval and are evaluated on facts instantiated at time points. The truth of a static predicate, e.g., a range condition for the variable, is tested for each time point in an interval, counting how often it is true. For *always* this needs to be

true for all time points, *sometimes* corresponds to least one success, and *true-percentage* can be parameterized as a compromise of the other two.

The Event logic defined in (Siskind, 2001) is a combination of Allen's relations with logical predicates that hold on intervals. The AMA (*and meets and*) logic used in (Fern et al., 2002) is a subset of the complete Event logic restricted to conjunctions of several meeting conjunctions of positive interval literals.

3.3 Temporal Concepts

Based on the various temporal operators used in the literature, we propose a unifying view of temporal concepts. A temporal concept is essentially an intuitive notion of the type of temporal relationship between time points or time intervals. A single temporal concept can correspond to several temporal operators. We give examples for each concept using the weather.

3.3.1 Duration

The concept of *duration* is the persistence or repetition of a property over several time points. Duration is what distinguishes time point from time interval data models. Time points express instantaneous information, e.g., lightning during a thunderstorm. According to the time resolution of our visual perception lightning is perceived as an instantaneous event without duration. The following thunder, however, usually lasts for a few seconds and can thus be described by an interval and has a duration. Time points will be displayed by vertical lines, time intervals as horizontal bars (see Figure 3.2).

3.3.2 Order

The concept of *order* is the sequential occurrence of time points or time intervals. A flash is always followed by a thunder in a storm, an arrow is used to represent the ordering between the two events in Figure 3.2. For both data models the concept of order corresponds to the operators *before* and *after*, and for intervals also the more restrictive *meets* operator.

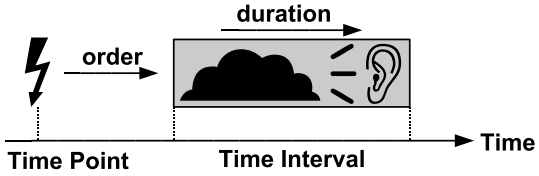


Figure 3.2: The temporal concept of duration is represented by an interval indicating the presence of thunder. The temporal concept of order is shown by consecutive placement of temporal events according to the direction of time from left to right.

3.3.3 Concurrency

The concept of *concurrency* is the closeness of two or more temporal events in time in no particular order. In a heavy thunderstorm lightning and drops of rain occur concurrently, but there is no special order relation between these events (Figure 3.3). Concurrency is often used for time points, especially for the data model of time sequences where the exact local order of time points is not necessarily meaningful. With intervals it correspond to the occurrence of two or more intervals within a larger sliding window and no further constraints on their relative position.

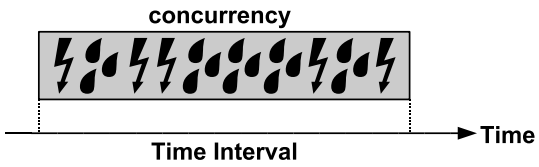


Figure 3.3: The temporal concept of concurrency represents closeness in time with no particular order. The time during which raindrops and flashes of lightning occur concurrently is shown as an interval.

3.3.4 Coincidence

The concept of *coincidence* describes the intersection of several intervals. If rain coincides with sunshine, there will often be a rainbow visible. Both rain and sunshine could have started earlier or last longer, but only when they coincide the rainbow is visible. The lifetime of the rainbow interval in Figure 3.4 thus corresponds to the overlap of the sunshine and rain intervals. The concept of coincidence is present the *contains* operator and nine of Allen's relations (*overlaps*, *starts*, *during*, *finishes*, the corresponding inverses, *equals*). The *coincides* operator directly expresses this temporal concept.

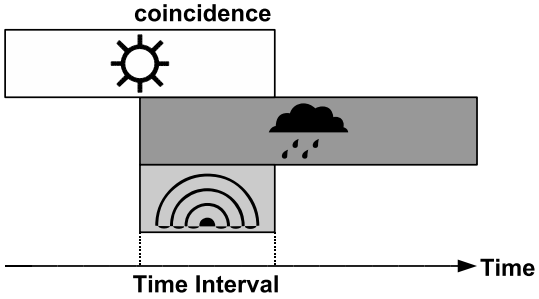


Figure 3.4: The overlapping parts of two or more intervals represent the temporal concept of coincidence. In the example rain, sunshine, and a rainbow coincide for a certain duration.

3.3.5 Synchronicity

The concept of *synchronicity* is the synchronous occurrence of two temporal events and is a special case of coincidence. The flash of lightning and the shrinking of our pupils to adjust for the brightness are synchronous time point events at the accuracy of our perception. For time intervals imagine a sunny spring afternoon with a cloud passing the sun. During this time interval it will be slightly darker and cooler. Both effects set in and end synchronously with the shadow approaching and receding. The corresponding intervals in Figure 3.5 are aligned. For both time points and intervals synchronicity mainly corresponds to the operator *equals*. For time intervals it can also be the relaxed form *more or less simultaneous* to allow for inaccuracies in the measurements of the interval boundaries. For time points such a threshold only makes sense for time sequences and it needs to be small, otherwise it would rather correspond to the concept of concurrency.

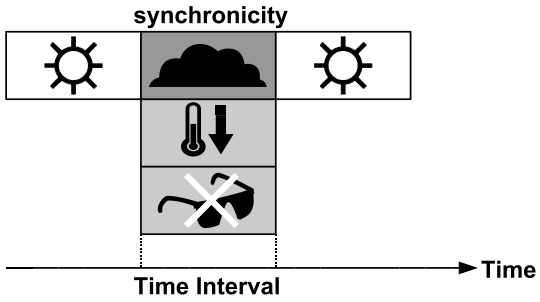


Figure 3.5: The temporal concept of synchronicity is a special case of coincidence. The involved intervals need to start and end simultaneously like the presence of a cloud and the change in temperature and brightness in the example.

3.3.6 Periodicity

The concept of (approximate) *periodicity* is the repetition of the same or very similar values with an (almost) constant period. A common example is the seasons we experience each year with a period of about 12 months.

3.3.7 Summary

In Figure 3.6 we show which temporal concepts are expressed by the time point operators and by Allen's time interval operators. The inverses of operators are not shown, because they correspond to the same temporal concept. The remaining interval operators are not shown because they correspond to disjunctions of Allen's operators. The four time point operators express the four most common temporal concepts used with time points. Coincidence for time points corresponds to synchronicity. Duration can be modelled with time points by immediate repetition of the same symbol and is implicitly present in intervals. Allen's interval relations cover the three most common temporal concepts used with intervals: order, coincidence, and synchronicity. Four different operators partly express coincidence. Periodicity and concurrency are not commonly used for intervals.

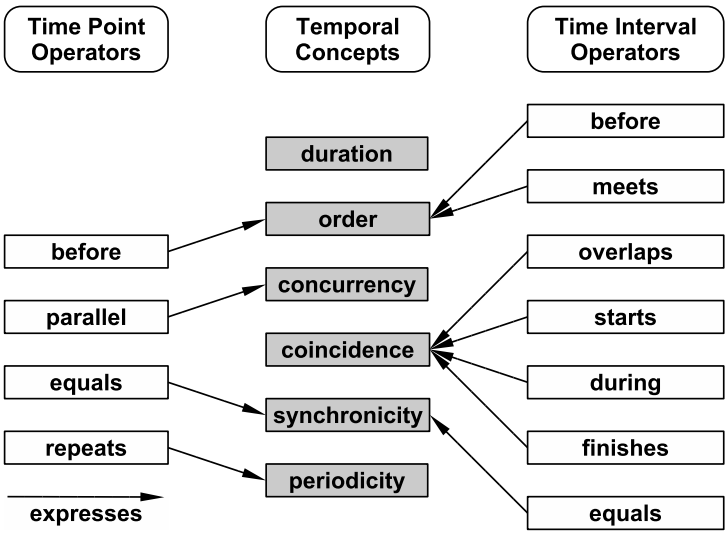


Figure 3.6: The outer columns list all time point and time interval operators. The arrows pointing to the temporal concepts in the center columns indicate which concepts are expressed by which operators.

Chapter 4

Temporal Data Mining

4.1 Introduction

We see temporal data mining (TDM) as the mining of data that has some temporal aspect to it. This covers a large area of problems and mining methods. Naively applying traditional data mining techniques by treating the temporal attributes as static features and ignoring the temporal structure of the data is usually not enough. Many special temporal mining algorithms have been developed for various temporal data types. Since temporal data mining is a relatively new field of research, there is no widely accepted taxonomy yet.

Chen and Petrounias propose a framework for temporal data mining. A temporal pattern is defined as tuple consisting of a static pattern and a temporal expression indicating when the pattern is valid (Chen and Petrounias, 1998).

Morik defines representation languages for temporal data mining and categorizes tasks by distinguishing two temporal concepts (linear precedence and immediate domination) and five data models (multivariate and univariate time series, nominal valued time series, sequence vector, and facts) (Morik, 2000). Within this framework they describe the experiences and results of three case studies.

Roddick and Spiliopoulou define paradigms for (Spatio-) temporal data mining. They distinguish three temporal data types (scalar, events, mining results) and two mining paradigms (Apriori-like, classification) (Roddick and Spiliopoulou, 2002).

Lin et al. describe temporal data mining as *"the analysis of temporal data and [...] finding temporal patterns and regularities"*. They list several mining tasks (classification, clustering, induction) and problems (similarity, periodical) (Lin et al., 2001).

The following sections describe the some important areas of temporal data mining and some related topics. Similar categorizations of temporal data mining can be found in (Dunham, 2002) and (Roddick and Spiliopoulou, 2002). Since time series are the main focus of this thesis, the large area of time series data mining (TSDM) will be explained in more detail in Chapter 5. Similarly, Chapter 6 is devoted to the particular task of unsupervised rule discovery from time series, listing approaches most relevant for this thesis. Figure 4.1 visualizes this upcoming zoom into temporal data mining.

4.2 Sequential Pattern Mining

The search of frequent itemset subsequences is commonly called *sequential pattern mining* (Agrawal and Srikant, 1995). A typical application is customers purchasing sets of items at different times, where the seller is interested in products typically bought some time after an initial purchase. This way recommendations can be given. Itemset sequences can be searched

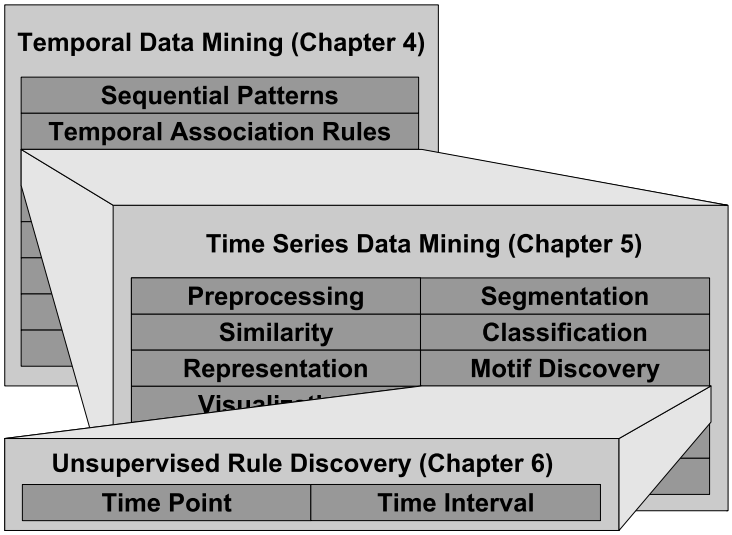


Figure 4.1: Chapters 4 through 6 describe the nested topics temporal data mining, time series data mining, and unsupervised rules discovery at increasing levels of detail.

with the well known AprioriAll (Agrawal and Srikant, 1995) algorithm. A faster method is SPADE (Sequential Pattern Discovery using Equivalence classes, Zaki (1998)). Generalizations of sequential patterns using concept hierarchies and temporal constraints were proposed in (Srikant and Agrawal, 1996). They also describe a fast algorithm called GSP (Generalized Sequential Pattern).

An algorithmically different approach for mining sequential patterns is called *pattern-growth* (Han and Pei, 2001) using, e.g., the PrefixSpan (Pei et al., 2001) algorithm. The pattern-growth approach uses the divide and conquer principle to avoid or dramatically reduce the generation and testing of candidate patterns. The SPAM (Sequential Pattern Mining, Ayres et al. (2002)) approach is also a depth-first search using a memory resident bitmap representation to achieve a high speedup compared to SPADE in general and to PrefixSpan for large data sets.

Recently, CloSpan (Yan et al., 2003) has been proposed to mine only closed sequential patterns, i.e., patterns with no super-patterns of the same frequency. To free the user from choosing the minimum frequency, Tzvetkov et al. propose to mine the top k closed patterns raising the threshold dynamically (Tzvetkov et al., 2003). In (Wang and Han, 2004) the Bi-Directional Extension checking (BIDE) algorithm is demonstrated to outperform CloSpan. The Apriori and PrefixSpan algorithms were extended to multidimensional sequential data in (Yu and Chen, 2005).

Episode rules (Mannila et al. (1997), Chapter 6) applied to itemset sequences are similar to sequential patterns. An Episode can represent a partial order of itemsets, an Episode rule describes an implication of Episodes.

4.3 Temporal Association Rules

While sequential patterns describe the concept of order in itemset sequences, temporal association rules combine traditional association rules with temporal aspects. Often the time stamps are explicitly used to describe the validity, periodicity, or change of an association.

Özden et al. mine association rules that hold only during certain cyclic time intervals (Özden et al., 1998). The authors argue, that when searching over several months, a co-occurrence of, e.g., *coffee* and *doughnuts* might have relatively low frequency. But when looking

at hourly sales figures they might be more frequent in the morning hours. The temporal granularity needs to be specified by the user.

Similarly, Chen and Petrounias combine association rules with temporal constraints on the validity (Chen and Petrounias, 1998). The temporal features are based on a user defined calendar and can describe an absolute time, a periodical time, or a specific time. The support is only measured during these time intervals. Algorithms to mine the rules given the temporal constraints (Chen et al., 1998) and to mine the valid time periods given the rules (Chen and Petrounias, 1999) are proposed.

In (Ale and Rossi, 2000) time is used for a more meaningful calculation of association rule support. The lifetime of an item is defined as the time between the first and the last occurrence. The temporal support is calculated w.r.t. this interval. This way rules are found that are only active during a certain time. In addition, outdated rules can be pruned by the user.

Inter-transaction association rules (Lu et al., 1998; Tung et al., 1999) merge all itemsets within a sliding time window augmented with the occurrence times. The resulting rules can express occurrences with relative time differences. Recently a faster algorithm has been proposed for this problem (Huang et al., 2004).

Rainsford and Roddick define temporal association rules as traditional association rules plus a conjunction of binary temporal predicates (see Chapter 3.2, Rainsford and Roddick (1999)). A separate confidence factor is assigned to the basic rule and each temporal operator. The temporal aspects of the rules can be visualized in a parallel coordinate plot (Rainsford and Roddick, 2000).

In (Dong and Li, 1999) emerging patterns are defined as patterns whose support increases significantly over time. It is argued, that these changes are often more interesting than rules with constant support that more likely indicate known facts.

In (Wijsen, 1997) the concept of trend dependencies is defined to express rules of the form: *"If an employee's rank increases then his/her salary does not decrease"*. Wang et al. analyze the evolution of numerical attributes over some time steps (Wang et al., 2001). They define temporal association rules as the correlations between two sets of evolving attributes.

4.4 Data Streams

Data streams are characterized by a continuous high rate flow of incoming data. Due to the speed and size of the stream, the data cannot be stored as persistent relations. One data element is thus seen only once, the retrieval of older records is prohibitively expensive. Babcock et al. describes the data stream model as: *"the data elements in the stream arrive online"*, *"the system has no control over the order [...]"*, *"data streams are potentially unbounded in size"* (Babcock et al., 2002) and there is only one chance of fast access for each data element. Some applications of data streams are financial data, network monitoring, web applications, and sensor networks. The fundamental research issues w.r.t. database queries are described in (Babcock et al., 2002). Domingos and Hulten describe a framework to perform data mining on streams meeting the following requirements: (An algorithm) *"must require small constant time per record [...]. It must use only a fixed amount of main memory [...]. It must be able to build a model using at most one scan of the data [...]. It must make a model available at any point in time [...]. It should produce a model that is equivalent (or nearly identical) to the one that would be obtained by the corresponding ordinary data mining algorithm [...]. When the data-generating phenomenon is changing over time [...] the model at any time should be up-to-date"* (Domingos and Hulten, 2003). The concepts are applied, e.g., to decision trees (Domingos and Hulten, 2000). An extension for adapting to concept drift, i.e., changes of the underlying model is described in (Hulten et al., 2001).

4.5 Web Usage Mining

Web Usage Mining is the analysis of web log data in search for typical access patterns of users. These patterns can be used to improve the website structure or prefetch pages likely to be accessed next. In e-commerce environments the identification of customers is an important issue. There are three major steps involved (Srivastava et al., 2000): preprocessing (reformatting log data, e.g., to identify user sessions), pattern discovery, and pattern analysis. The above mentioned sequential patterns as well as Episode rules can be applied to web log data. Based on typically available background knowledge other specialized algorithms have been developed. The mining of so called Maximal Frequent Forward Sequences involves removal of backward traversals and page refreshes. In contrast Maximal Frequent Sequences do not exclude the steps back because they can give important hints for changing the site structure, e.g., by additional shortcut links (Dunham, 2002).

Pattern analysis is distinguished from pattern discovery by a more interactive mining process involving user specified patterns. MiDAS (Mining Internet Data for Associative Sequences) extends traditional sequential pattern discovery with the following web-specific features: navigation templates to guide the search, incorporation of the website topology, concept hierarchies, e.g., for categorizing referrer URLs, and application dependent syntactic constraints (Baumgarten et al., 1999). For the WUM framework (Spiliopoulou and Faulstich, 1999) a flexible query language, called MINT, has been developed. The pages in a pattern need not be accessed contiguously and wild cards are allowed.

4.6 Pattern Evolution

If one observes results from data mining algorithms at different points in time, interesting temporal developments can occur. Existing mining results can be updated with new data. These temporal data mining approaches are called *rule evolution* and *rule maintenance* (Roddick and Spiliopoulou, 2002). Maintenance and change detection of association rules is a common example (Cheung et al., 1997). In (Chakrabarti et al., 1998) unexpected changes in association rules over time are searched. A framework for mining the mining results, called *higher order mining*, is proposed in (Spiliopoulou and Roddick, 2000). The authors claim that the higher order semantics correspond to many everyday phenomena and give examples. A rule can change over time by differing in the antecedent, the consequent, or in the interestingness measures. Cotofrei and Stoffel compact a set of temporal rules by pruning with a confidence bound and merging rules based on the Minimum Description Length (MDL, Rissanen (1989)) principle (Cotofrei and Stoffel, 2002b, 2005).

4.7 Temporal Reasoning

Temporal reasoning deals with deduction algorithms based on logical representations of temporal phenomena. Scenarios are described with temporal constraints (see also Section 3.2). The consistency can be determined and queries about scenarios satisfying all constraints can be answered, see (Schwalb and Vila, 1997) for a survey. Inference about past, present and future is performed to support applications in planning, understanding, and diagnosis. Some important issues include the identification of tractable subclasses of constraint algebras (Nebel and Bürckert, 1994), enhancement of exact search algorithms, and development of polynomial time approximations (Schwalb, 1998).

4.8 Temporal Databases

Temporal databases are specialized database system that offer functionality for temporal queries. They support the temporal data mining process but they are not needed to pursue this goal (Roddick and Spiliopoulou, 2002). In (Böhlen et al., 1998) data models for temporal databases are discussed. (Ozsoyoglu and Snodgrass, 1995) provide a survey on data models and query languages, including real-time databases. A glossary of temporal database concepts is available in (Jensen et al., 1994).

Chapter 5

Time Series Data Mining

5.1 Introduction

As time series data mining we consider research dealing with data mining using numeric and symbolic time series and sequences. We also include the interval data types, because they are commonly obtained from time series or time sequences.

A brief overview of time series data mining[1] is given in (Antunes, 2001). They identify the representation of time series data and the definition of a suitable similarity measure as the main problems. The problems of outliers, noise, and scaling differences need to be addressed. They consider two different data models: numeric and symbolic time series. Accordingly, they list representation methods with common similarity measures and give some examples of applications to clustering, classification, and prediction. Finally, they comment that *"the research has not been driven so much by actual problems but by an interest in proposing new approaches"* (Antunes, 2001). Lin et al. consider the following tasks as the major time series data mining areas: subsequence matching, anomaly detection, and motif discovery (Lin et al., 2004). Earlier work by the same authors also lists indexing, clustering, classification, segmentation (Keogh and Kasetty, 2002) and summarization (Lin et al., 2003a).

In this chapter we present an extended list of typical tasks for time series data mining. This includes methods for converting between the data models, e.g., a numeric time series into a symbolic interval sequence. Many examples from the literature are pointed out for each data model where the task is applicable. We conclude with a discussion of the state of the art in time series data mining with respect to the goal of discovering understandable knowledge. The main focus of this thesis, unsupervised discovery of rules from time series, is deferred to Chapter 6.

5.2 Preprocessing

Preprocessing of time series data requires special methods to incorporate the temporal dimension. We will list the most important techniques briefly and give some examples. Many other application dependent methods are possible.

A very important step when using time series obtained from real life measurements is the reduction of noise. Noise filtering can be done, e.g., using digital filters (Smith, 1997) or wavelet thresholding (Jansen, 2001).

For methods requiring stationary signals, the trend needs to be removed. This can be done by fitting linear or higher order functions and keeping the residuals. Differencing is a very simple and often effective method for trend removal, but noise is amplified and the meaning

[1] They call it temporal data mining.

of the feature is changed. Missing values can often be replaced by linear interpolation. For financial time series the last available value is commonly used.

Feature extraction techniques can be used to convert a series of original values to a more meaningful domain or to provide additional information. They can return a time series, possibly with a lower resolution, a single value, or a static vector of features. In (Himberg et al., 2001b) principal component analysis (PCA, Jolliffe (1986)) and independent component analysis (ICA, Hyvärinen (1999)) are used to preprocess sensor data. For musical data instantaneous frequency (Goto, 2004) or other summaries of the short term frequency content are used, see (Mörchen et al., 2005) for an overview. For financial time series the log of the first order differences is commonly used (de Bodt et al., 2001). A time series measuring the instantaneous probability of a change point is used in (Idé and Inoue, 2005) to analyze the correlation between heterogeneous variables.

Feature extraction can also change the data model. In (Mannila and Salmenkivi, 2001; Vlachos et al., 2004b) time series describing the local frequencies of symbols in a symbolic sequence are analyzed.

5.3 Similarity

The notion of time series similarity is of utmost importance for many other mining tasks, e.g., clustering. We will mainly use the dual term distance.

5.3.1 Numeric Time Series Distances

There are four categories of similarity measures for numeric time series (Keogh, 2004). Shape-based methods compare the overall appearance of the time series. Feature-based methods extract features that usually describing time independent aspects of the series that are compared with static distance functions. Model-based methods fit a model to the data and measure the similarity by comparing the models. Compression-based methods analyze how well two time series can be compressed alone and together. The categories are shown in Figure 5.1 with some examples explained below.

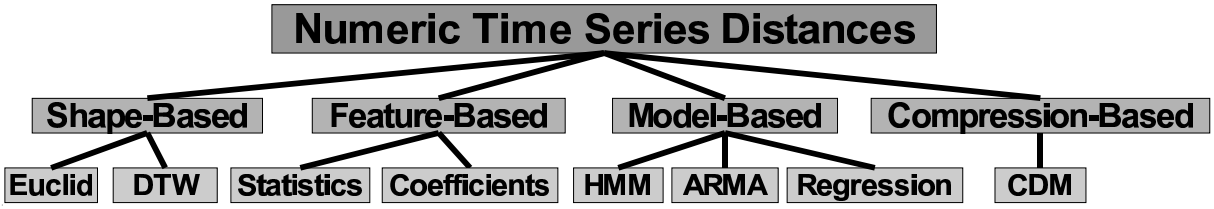


Figure 5.1: Categorization of numeric time series distances. Short time series are commonly compared by shape. Features or models can be derived and compared for long time series. The compression-based distance measures how well the concatenation of two series can be compressed.

The Euclidean distance is the most widely used shape-based distance for the comparison of numeric time series (Keogh and Kasetty, 2002). More generally, other L_p norms (Yi and Faloutsos, 2000), i.e., Manhattan for $p = 1$, Euclidean for $p = 2$, Maximum for $p = \infty$, can be used as well, putting different emphasis on large deviations. There are several pitfalls when naively using L_p distances on time series: scaling and translation of the amplitudes or the time axis, noise and outliers, uniform and non-uniform time warping (e.g. Antunes (2001); Keogh and Kasetty (2002)). Therefore many authors have used transformations that make the

Euclidean distance invariant to these differences. Care has to be taken in choosing the transformations to obtain a time series distance measure that is meaningful to the application.

A translation or scaling of the amplitude can produce a large distance between time series even if they have a similar shape. A common example where this is not desired is the comparison of stock prices. The absolute value of a stock is usually not as interesting as the shape of up and down movements. This problem can be overcome by a linear transformation or normalization of the amplitudes (Goldin and Kanellakis, 1995). Normalizing to a fixed range (Agrawal et al., 1995a) or zero mean and unit variance (Keogh and Kasetty, 2002) needs to be applied to both time series. These methods are fast and very common, but they do not give the optimal match of two series under linear transformations (Kahveci et al., 2002; Argyros and Ermopoulos, 2003). A linear transformation should only be applied to one of the time series, because trying to find two transformations simultaneously always gives zero scaling factors as the optimal (and useless) answer (Goldin et al., 2003). For querying of time series databases (see Chapter 5.12) some authors transform the query (Chu and Wong, 1999), others transform the candidate series (Goldin and Kanellakis, 1995). The minimum of both possibilities is used in (Kahveci et al., 2002). In (Goldin et al., 2003) the transformation is searched with optional bounds on the amount of scaling and shifting. The scaling factor, in particular, should not be allowed to be negative for many applications, because this corresponds to an inverted shape.

Both normalization and linear transformation should be handled with care in the presence of noise (Vlachos et al., 2002). If one time series is essentially flat with some noise, normalizing it to unit variance will make it look very different. Noise can also be emphasized by linear transformations in order to match features of the other time series. Goldin et al. give examples where similarity after normalization does not correspond to qualitative similarity (Goldin et al., 2003).

Another problem with L_p distances of time series is the presence of outliers or noisy regions. This can be compensated by allowing gaps in the matching of two time series. An early approach matches short sub-series with gaps in between to determine similarity (Agrawal et al., 1995a). No distance function is defined, however. Das et al. align two time series using the longest common subsequence (LCSS) algorithm (Das et al., 1997). A linear transformation is allowed, two points are compared using a threshold. The similarity is defined as the length of the LCSS. Fast approximate algorithms are described in (Das et al., 1997; Bollobas et al., 2001). Vlachos et al. normalize the LCSS similarity by the length of the time series, allow linear transformations, and define distance functions (Vlachos et al., 2002, 2004a).

Scaling and translation of time is usually not a big problem because the time axis values are not commonly used in distance calculation, but rather the order of values. Problems arise however with time series of different lengths and/or sampling rates. Resampling can be used to make both time series of the same length and then apply a distance function (Keogh and Kasetty, 2002), sometimes referred to as uniform time warping (Palpanas et al., 2004). Down-sampling the longer series has been observed to be efficient and robust against noise (Argyros and Ermopoulos, 2003).

Small distortions of the time axis are commonly addressed with non-uniform time warping (Keogh and Pazzani, 1999), more precisely with Dynamic Time Warping (DTW, Berndt and Clifford (1996)). The DTW distance allows warping of the time axes in order to align the shapes of the two times series better. The two series can also be of different lengths. The optimal alignment is found by calculating the shortest warping path in the matrix of distances between all pairs of time points under several constraints. The point-wise distance is usually the Euclidean or Manhattan distance. The DTW is calculated using dynamic programming with time complexity $O(n^2)$. See (Ratanamahatana and Keogh, 2004b) for a concise description of DTW. The same authors have recently demonstrated that using a lower bounding technique (Keogh, 2002) the complexity is essentially $O(n)$ (Ratanamahatana and Keogh,

2004a). Salvador and Chan describe an approximation algorithm with time complexity $O(n)$ (Salvador and Chan, 2004).

Two distance measures related to L_p distances and sometimes used with static data are the correlation and the cosine distance. The former is based on the Pearson correlation coefficient that measures linear dependence. The time series need to be of the same length and all time points are assumed to be independent samples from the same probability distribution. The estimate of the correlation may be poor when the series are short. Examples can be constructed where the distance does not correspond to the intuition of shape similarity (Möller-Levet et al., 2003). The correlation distance is sometimes used with financial time series (Struzik and Siebes, 1999; Ormerod and Mounfield, 2000), that are commonly assumed to be close to random walk (de Bodt et al., 2001). In (Bagnall et al., 2003) correlation is used to compare a discretized representation to the numerical series in the context of clustering. The algorithm for retrieving matches to a visual query in (Haigh et al., 2004) also uses correlation. The cosine distance measures the angle between the time series interpreted as a vector in a high dimensional space and is equivalent to the correlation distance if the mean is subtracted from each time series.

Many of the proposed alternatives to the Euclidean distance are not superior. Keogh and Kasetty have evaluated 11 different measures (some of which are explained above) on two time series classification benchmark data sets. The results obtained with the Euclidean distance were an order of magnitude better than those of any other measure. We refer the interested reader to (Keogh and Kasetty, 2002) and the references therein for the remaining distance functions.

For long time series of possibly very different lengths, the shape-based methods do not give intuitive results, feature- and model-based methods should be considered. The method of extracting features from the time series and comparing the feature sets is very domain dependent. If only certain properties of the time series are important, the right features will give good results. One example of rather general features is the use of the first four moments of the empirical probability distribution of the data and the first order differences (Nanopoulos et al., 2001). Additional features based on trend, seasonality, and self-similarity were used in (Wang et al., 2004). For a set of time series globally important features can be selected using the most important coefficients from Discrete Fourier Transform (DFT, Shatkay (1995b)) or Discrete Wavelet Transform (DWT, Mallat (1999)) decompositions (Mörchen, 2003). In (Zhao et al., 2005) the coefficients are chosen independent from the data by considering a weighting of the time points with a bias towards recent values. For the DFT representation a likelihood ratio distance has been shown to outperform Euclidean distance of the coefficients on data where the low frequencies do not have discriminatory power (Janacek et al., 2005). In (Vlachos et al., 2005) the most important periods are selected combining the periodogram with the autocorrelation function. These examples are termed *Statistics* and *Coefficients* in Figure 5.1.

Compared to feature-based methods, model-based distances have the additional advantage, that prior knowledge about the data generating process can be incorporated. The similarity can be measured by modelling each time series and determining the likelihood that one series was produced by the model from another series. The average of the pairwise likelihoods can be used to obtain a symmetric similarity measure. For numeric time series Hidden Markov Models (HMM, Rabiner (1989)) with continuous output values or Auto Regressive Moving Average (ARMA) models (Box et al., 1994) are common choices (e.g. Smyth (1997); Perrone and Connell (2000); Xiong and Yeung (2003)). In (Ge and Smyth, 2000) HMM models are combined with a piecewise linear representation. In (Panuccio et al., 2002) the distance between HMM models is normalized to take into account the goodness of fit of the series producing the model. Alternatively to the likelihood distance, the models of two series can be compared directly, e.g., for ARIMA models using the model parameters (Deng et al., 1997) or derived coefficients (Kalpakis et al., 2001).

Qian et al. compare the transition matrices of HMM obtained from clustering the phase space of time series (Qian et al., 2003).

The above models need long time series for a good estimation of their parameters. For short smooth time series with varying lengths and irregular sampling Gaffney and Smyth propose to use regression models (Gaffney and Smyth, 1999, 2003). Since the regression models represent the shape of the time series with a function, this is a hybrid approach of model-based and shape-based representations.

An approach that uses ideas from shape and feature-based representations is described in (Megalooikonomou et al., 2005). Typical local shapes are extracted with vector quantization (Gersho and Gray, 1992) and the time series are represented by histograms counting the occurrences of these shapes at several resolutions. The weighted distance between the histograms determines the time series similarity.

The compression-based similarity of time series is a very recent idea. Inspired by results in bioinformatics and computational theory, Keogh et al. define a distance measure based on the conditional Kolmogorov complexity (Keogh et al., 2004b) called Compression-Based Dissimilarity Measure (CDM). The Kolmogorov complexity is the length of the shortest program that is able to generate the given data. This can be approximated by assuring that the data is in a format suitable for compression and using the best compression algorithm available. The basic idea is, that concatenating and compressing similar data should give higher compression ratios than doing so with very different data. For time series they proposed to use discretization before compression. The presented results in clustering, anomaly detection and classification are excellent. The method should only be applied to long time series, but similar to model-based methods, they are allowed to have very different lengths.

Note that the compression-based method and the model-based likelihood methods only give similarity or distance values between pairs of time series. They cannot be used to compute prototypical time series that represent an average or centroid between two or more series. This excludes the application of data mining methods designed to work in vector spaces, e.g., k-Means, Self-Organizing Maps (SOM), Learning Vector Quantization (LVQ), see (Kohonen, 1995). For model-based methods that compare the parameters of the models, it is not obvious whether the average of two or more parameter sets will adequately represent the notion of an average model that is useful as prototype for the other models.

Simple guidelines for choosing a distance for time series data mining problems are (Keogh, 2004): If the time series are relatively short, of similar lengths, and shape is a meaningful description, DTW should be used. If the time series are long and few knowledge about the structure is available, CDM should be tried. If prior knowledge about the structure of the data is available, feature-based or model-based methods may provide a more meaningful abstraction and better results.

5.3.2 Symbolic Time Series Distances

For series of symbols we identified four types of distances: proximity-based, feature-based, model-based (Bicego et al., 2004), and compression-based (Keogh et al., 2004b). The categories are shown in Figure 5.2 with some examples explained below.

Proximity-based methods measure the distance between two series directly from the symbolic representation, e.g., by the Hamming distance (e.g. Gusfield (1997)), that simply counts the number of positions where the sequences differ. The family of edit distances is motivated by string matching. The similarity between two symbolic series is measured by determining the cost of transforming one into the other. The edit operations are insertion, deletion, and substitution of a symbol and can be given different costs. With unit costs for all three operations, the Levenshtein distance is obtained. Token-based distances are based on multisets of the

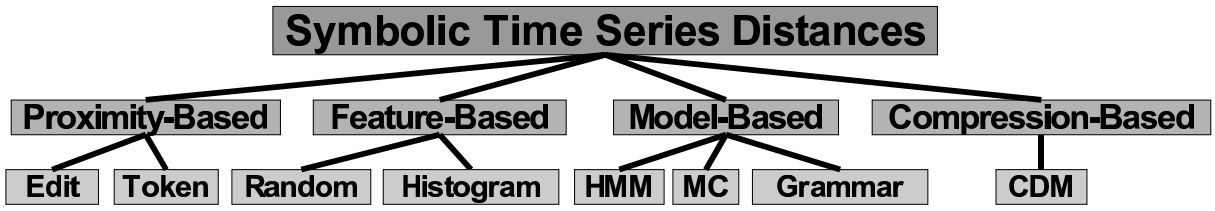


Figure 5.2: Categorization of symbolic time series distances. Proximity is measured by determining the cost of transforming one series into another. Features or models can be derived and compared for long time series. The compression-based distance measures how well the concatenation of two series can be compressed.

symbols (or short sub-series extracted with a sliding window) of the two series, thus loosening or ignoring the order of the symbols. The Jaccard similarity is calculated by dividing the size of the intersection by the size of the union of the two multisets. See (Cohen et al., 2003) for more information on string matching distances, including hybrids of edit and token based measures.

Feature-based methods convert each symbolic series to a numeric feature vector capturing the temporal information. These vectors can then be compared with the Euclidean distance or similar distance functions. Mannila and Seppänen represent each symbol by a high dimensional numerical random vector and a symbolic sequence by the sum of the vectors weighted by the temporal distance of the symbols (Mannila and Seppänen, 2001). In (Flanagan, 2003) weighted symbol histograms of consecutive symbols are used as features. These examples are termed *Random* and *Histogram* in Figure 5.2.

Common choices for the model-based approach are Markov Chains (MC) (Reinert et al., 2000; Ridgeway, 1997), HMM with discrete output distributions (Law and Kwok, 2000), and grammar based models (Antunes, 2001). When using grammars the output is binary, depending on whether a time series is recognized by a grammar or not. The other models return probability values. Alternatively to calculating pairwise likelihood of two series and the corresponding models, the Kullback Leibler divergence (Kullback and Leibler, 1951) can be used to compare the models directly, e.g., for MC (Sebastiani et al., 1999; Ramoni et al., 2002).

Since the compression-based method for numeric time series includes discretizing the data, the CDM (Keogh et al., 2004b) measure can also be used directly with symbolic time series.

The guidelines we want to give for choosing a symbolic distance are similar to those for numeric time series. Using proximity-based methods requires the concept of, e.g., an edit distance to be meaningful which will often be the case for short symbolic series. For long series, CDM can be used in lack of knowledge, while feature- and model-based approaches may be justified by the application domain.

5.4 Time Series Representation

The main motivation behind representing a time series in some form other than using the actual values is to better represent the main characteristics of the data in a concise way. Additional benefits can be compression and thus speedup of processing as well as noise removal and emphasis of important features. Often, the quality criterion is the reconstruction error aiming to represent the original time series well. The selection of a suitable time series representation is interconnected with the selection of a distance measure. Many common representations support the approximate calculation of the Euclidean distance of the original time series. Only few support the weighted Euclidean distance (Keogh et al., 2001b). For numeric time series a taxonomy inspired by (Lin et al., 2003a; Keogh, 2004) is shown in Figure 5.3. There are few alternative

representations for symbolic time series. We will mention them in the categories where they fit best, but excluded them from the figure.

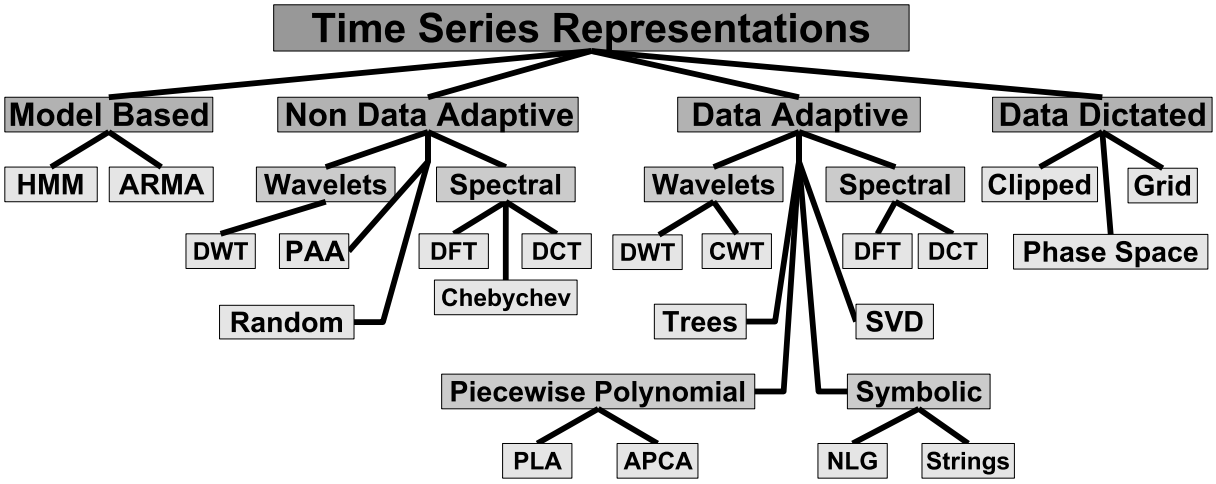


Figure 5.3: Categorization of time series representations. Models can be used to capture the structure of the data generating process. Non data adaptive have a fixed size and the comparison of representations from several time series is straightforward. Data adaptive methods also have a fixed size. They can better approximate each series, but the comparison of several series is more difficult. The size of data dictated representations depends on the data.

5.4.1 Model-Based Representation

The model-based representations assume that the time series have been produced by a certain model. The models are fitted to the data and the resulting parameters of the models are used for further mining tasks. A simple and common model for symbolic time series is a MC (Sebastiani et al., 1999). A more flexible way of modelling is offered by HMM that can be used for symbolic and numeric time series. HMM with numeric outputs are used in (Smyth, 1997) and (Panuccio et al., 2002). Another popular choice for numeric time series is statistical modelling with ARMA models, e.g., (Sebastiani and Ramoni, 2001; Kalpakis et al., 2001; Xiong and Yeung, 2003). The statistical features extracted in (Nanopoulos et al., 2001) and (Wang et al., 2004) can also be interpreted as a model of the process generating the time series.

5.4.2 Non Data-Adaptive Representation

The non data adaptive models transform the time series into a different space while the transformation and the selection of a subset of the coefficients are independent of the data. Both wavelet and spectral representations express numeric time series as coefficients in a function space spanned by a set of basis functions. The DWT uses scaled and shifted versions of a mother wavelet function, usually with compact support, to form an (bi-)orthonormal basis. This gives a multi-resolution decomposition where low frequencies are measured over large intervals and high frequencies over small intervals. A large number of basis functions exists, the most prominent and simple is the Haar wavelet, other popular choices are Daubechies, Symlet, and Coiflets (Mallat, 1999). For applications in time series indexing see (Chan and Fu, 1999) and (Popivanov and Miller, 2002). The DFT uses complex exponential functions, representing sine and cosine functions in the real domain and is used in (Agrawal et al., 1993a) and (Rafiei and Mendelzon, 1998). The Discrete Cosine Transform (DCT) only uses cosine functions.

It has not been used extensively in the literature and does not seem to offer any advantages over the DFT (Keogh, 2004). Using the first coefficients of the DFT or DWT corresponds to keeping the low frequency components of the time series and dropping high frequencies assumed to be noise. Keeping only a small fraction of the coefficients leads to high compression ratios with low reconstruction errors if the signals really do mainly contain low frequencies. Otherwise data-adaptive methods should be used (Mörchen, 2003). Chebychev polynomials (Cai and Ng, 2004) provide a low order polynomial approximation to polynomials that are minimal in the maximal deviation. This representation can also trivially handle multivariate time series. The Piecewise Aggregate Approximation (PAA, Keogh et al. (2001b)) also known as Segmented Means (Yi and Faloutsos, 2000) represents a numeric time series by the mean values on consecutive fixed length segments. Analogous to DWT and DFT this can be interpreted as transforming it into a space spanned by box basis functions. An extension to multi-resolution PAA (MPAA) is proposed in (Lin et al., 2005). Random projection (Dasgupta, 2000) of time series is used for representation in (Indyk et al., 2000). Each time series is convolved with k random vectors drawn from a multivariate standard normal distribution of the same dimensionality. the resulting vector of k checksums is called a sketch.

5.4.3 Data-Adaptive Representation

Data adaptive methods depend either locally on the values of one time series or globally on the set of all time series. Many of the decompositions described above can be used in a data adaptive manner by choosing the largest coefficients instead of the first coefficients. For DFT this corresponds to keeping the most dominant frequencies and is optimal in the sense of energy preservation (Wu et al., 2000). This principle can be applied locally, for each time series separately (Vlachos et al., 2004b), or globally for a set of time series (Mörchen, 2003). An inherently data adaptive wavelet representation based on the Continuous Wavelet Transform (CWT) and the Wavelet Transform Modulus Maxima (WTMM, Mallat (1999)), respectively, is presented in (Struzik and Siebes, 1999). The largest singular features of a time series are extracted and represented by their position, the local roughness, and the sign of the wavelet coefficient.

Singular Value Decomposition (SVD, Jolliffe (1986)), the main processing step in PCA, is the optimal linear transform in the sense of energy preservation. In the context of time series the eigenvectors are commonly called eigenwaves. SVD is used in (Korn et al., 1997) to represent time series, an extension to outlier handling is also proposed. In (Ravi Kanth et al., 1998) a fast method to recompute the SVD upon the arrival of new data is described.

A relational tree representation is used in (Shaw and DeFigueiredo, 1990). Non-terminal nodes of the tree correspond to valleys and terminal nodes to peaks in the time series. The tree is invariant to monotonic axis transformations and used to perform classification. Similarly, Bakshi and Stephanopoulos propose to extract pivotal temporal features using wavelet multiscale filtering for segmentation (see Chapter 5.8) and represent each interval by one of seven primitives describing the curvature of the segment (Bakshi and Stephanopoulos, 1995). An extended the set of 13 shapes is used in (Colomer et al., 2002).

For piecewise polynomial models the signal is segmented (see Chapter 5.8) and a polynomial is calculated per segment. Piecewise Linear Approximation (PLA) is commonly used by either performing interpolation (Shatkay and Zdonik, 1996; Keogh and Pazzani, 1998) or regression (Morinaka et al., 2001) to obtain a linear trend. Constant approximations per segment were independently proposed in (Geurts, 2001) and (Keogh et al., 2001a), called Adaptive Piecewise Constant Approximations (APCA) by the latter. A data adaptive version of PAA is proposed in (Megalooikonomou et al., 2004). Instead of using the mean on fixed length segments, vector quantization is used to create a codebook and each segment is represented

by the index of the closest codebook vector. The idea is extended to multiple resolutions in (Megalooikonomou et al., 2005).

Symbolic methods convert numeric time series to symbolic time series or even textual descriptions. In the context of knowledge discovery this can be a crucial step, therefore we will put special emphasis on the interpretability of the obtained symbolic representation. The most interpretable form of symbolic description is textual description, obtained by Natural Language Generation (NLG) in (Boyd, 1998) and (Sripada et al., 2003b). The TREND (Boyd, 1998) system uses the scale space representation of the CWT to identify important trends in univariate time series. The most important trends are those persisting over multiple scales. The resulting descriptions are found to be similar to the ones given by experts. Sripada et al. observe that the changes they needed to make to standard data processing methods in order to better describe time series to humans corresponded to the Gricean Maxims of quality, quantity, relevance, and manner. For example, interpolation instead of regression is used to estimate the slope of a time series segment. This way truly observed values are reported for interval boundaries. For some applications relevant patterns like a spike or a step (Yu et al., 2002) have been identified by experts and were used in further processing. To make the descriptions brief and orderly, unchanged states states or repetitions are not reported, very close time points are not distinguished, and reports are ordered consistently either by time or by attributes. The resulting descriptions are further processed by NLG systems to produce a textual description of time series (Sripada et al., 2003b). The concept is applied in the areas of weather forecasts (Sripada et al., 2003a), process control (Yu et al., 2002), and intensive care (Sripada et al., 2003c).

The conversion of a numeric time series to a symbolic time series can also be seen as a form of noise removal by complexity reduction. Overviews of supervised and unsupervised discretization methods for static data are given in (Dougherty et al., 1995) and (Kohavi and Sahami, 1996). Daw et al. discuss methods geared towards time series data (Daw et al., 2003). We will concentrate on unsupervised methods, that can be mapped to linguistic descriptions like *high* or *low* and where the symbols correspond to fixed length segments of the numeric time series.

The most obvious method to assign symbols to single values is range partitioning (Daw et al., 2003), e.g., by histograms. Equal width and equal frequency histograms are the most commonly used value based discretizations. The former method chooses equally spaced cuts between the minimum and maximum values, each bin of the latter contains almost the same number of data points. The cut points of the histogram can also be determined by statistics like the mean plus/minus one or two standard deviations or other application dependent choices, like certain percentiles in (Tino et al., 2000). In case of multi modal distributions of the time series values all these methods may not give intuitive results, because the boundaries of the histogram bins could be placed in high density regions. An estimate of the probability density approximation and Gaussian mixture models were used to manually obtain meaningful intervals in (Mörchen et al., 2006). This is cumbersome and not feasible for a large amount of data sets that show different characteristics, however. The PERSIST algorithm (Mörchen and Ultsch, 2005b) automatically chooses bins optimizing the persistence of the resulting symbolic states.

The symbolic aggregate approximation (SAX) (Lin et al., 2003a) uses equal frequency histograms on sliding windows to create a sequence of short words of symbols. Repeating words are dropped. The key feature of this method is the lower bounding approximation to the Euclidean distance, useful for indexing (see Chapter 5.12).

In (Chmielewski and Grzynala-Busse, 1996) the histogram bins are adjusted trying to decrease the entropy at the boundaries. Another approach to tackle the problem is using clustering and vector quantization (Dougherty et al., 1995). In (Giles et al., 2001) a one dimensional SOM is used, where each map neuron correspond to one output symbol. The topological ordering of the symbols can be of advantage for further processing.

Flexer and Bauer discretize multivariate time series using k-Means for vector quantization

and perform a one dimensional Sammon's mapping (Sammon, 1969) to obtain an ordering of the symbols.

Using short segments of the time series, higher order descriptions can be obtained, e.g., the trend by fitting a linear function. This is similar to PLA, but the segments are all of the same length and given a priori. In (Agrawal et al., 1995b) the shape definition language (SDL) is defined using the discretized slope values obtained from connecting consecutive values as terminal symbols. This was generalized in (Qu et al., 1998) to lines on unit length segments with user specified width.

In (Das et al., 1998) the discretization symbols represent short primitive shapes found by clustering segments of the time series obtained using a sliding window (see also Chapter 5.7.2).

The recently proposed time series bitmaps (Kumar et al., 2005) further compress a symbolic time series into a bitmap where each pixel corresponds to the frequency of a fixed length subsequence (see also Chapter 5.5).

5.4.4 Data-Dictated Representation

For data-dictated representations you cannot choose the level of detail, the size of the representation is automatically determined (Ratanamahatana et al., 2004, 2005). The grid-based representation (An et al., 2003, 2005) places a two dimensional regular grid over the time series. The value axis is quantized into a fixed number of equal width bins, the time axis bins correspond to the time points. Values that are in the same bin or within a small error tolerance of the last value's bin are omitted. The final representation is a bit string describing which values were kept and which bins they were in. For reasonable smooth time series a reduction of 90% is reported.

Discretizing a time series to a binary string is called clipping. In (Bagnall et al., 2003) this is done using the median as the clipping threshold and the representation is compressed using run length encoding at the bit level. The main advantage of this representation is the scalability to huge data sets due to low memory requirements and fast processing. It has also been shown to be robust against outliers (Bagnall and Janacek, 2005). The mean is used as a threshold in (Ratanamahatana et al., 2005).

The phase space representation (Takens, 1981) of a signal is a method from nonlinear time series analysis (Kantz and Schreiber, 1997). It offers a reconstruction of the state space of the dynamical system producing the observed time series. The time series is transformed into a set of vectors. If the time series is $\{x_i | i = 1, ..., n\}$ then the phase space vectors are $\{(x_i, x_{i+l}, ..., x_{i+(d-1)l}) | i = 1, ..., n - (d-1)l\}$ where l is the lag and d the dimension of the embedding. The dimension of the phase space can be chosen with the false nearest neighbor method (Kennel et al., 1992) while the lag is suggested to be chosen from first minimum of the auto-mutual information curve (Kantz and Schreiber, 1997). The phase space representation is larger than the original time series. Further transformation can be used to achieve a more compact representation, for example using Gaussian mixture models (Lindgren et al., 2004; Povinelli et al., 2004; Zimmerman et al., 2003). In (Bauer et al., 1998; Morik et al., 2000) a two dimensional phase space representation is used to detect outliers and level shifts.

5.5 Visualization

Visualization is an important part of data mining because humans have remarkable abilities to spot hidden patterns (Shneiderman, 1996). Surprisingly, there has not been a lot of research effort in the data mining community targeted to visualization of time series. The most common form is a line graph where the horizontal axis represents the time. An early approach linked time series clustering results with a calendar based representation to show patterns in daily

power consumption (van Wijk and van Selow, 1999). Hochheiser developed the paradigm of Timeboxes to visually and interactively specify a query against a large time series database. A similar approach for a single long time series is presented in (Haigh et al., 2004). A tool for linking spatial time series with other plots is described in (Andrienko et al., 2004).

The periodicity of time series can be visualized using spirals (Weber et al., 2001). The spiral display can reveal periodic structure much better than linear graphs. The value of a time series can be shown on the spiral using color or line thickness. Intertwined spirals can represent multivariate data.

The visualization of continuous dynamic systems with pipe diagrams is proposed in (Alonso and Rodriguez, 1999). The visualization of different regimes in the system as a directed acyclic graph is said to be end user compatible.

The Viz-Tree (Lin et al., 2004) is a tree showing all fixed length subsequences of a symbolic time series obtained by SAX discretization. It is shown to be useful for various time series data mining tasks, namely interactive (sub-)sequence matching (Chapter 5.12), motif discovery (Chapter 5.10), and anomaly detection (Chapter 5.11). The tree is linked with conventional time series displays to display the original values of the patterns selected by the user. The same discretization is used in (Kumar et al., 2005) and combined with the chaos game representation (Jeffrey, 1992) to generate small bitmaps for time series. The bitmaps can be integrated in standard file browsers and offer a similarity based visualization of time series files (Kumar et al., 2005).

5.6 Prediction

Prediction is usually understood as forecasting the next few values of a numeric or symbolic series. For the prediction of numeric time series there is a huge amount of literature especially in the field of statistics. Common methods for time series modelling and prediction are ARIMA (Auto Regressive Integrated Moving Average) (Box et al., 1994) models for stationary processes and GARCH (Generalized Autoregressive Conditional Heteroskedasticity) (Brockwell and Davis, 2002) models for processes with non-stationary variance.

More simple techniques like regression or more complicated techniques like supervised neural nets (Tang and Fishwick, 1991), unsupervised neural nets (Koskela, 2003), or Support Vector Machines (SVM) (Burges, 1998; Müller et al., 1997) can also be used. In (Armstrong, 2001) many different approaches and aspects of forecasting are treated. Recently, a hybrid technique combining clustering and function approximation per cluster has been proposed in (Sfetsos and Siriopoulos, 2004).

The prediction of symbolic time series or sequences is sometimes called classification. Similar to numeric time series the underlying process, producing the series of symbols is modelled and the model is used to predict the next outcome. MC and HMM are commonly used. Techniques from association rules (Lu et al., 1998) or recurrent neural networks and grammatical inference (Giles et al., 2001) have been used to predict the movement of financial symbolic time series obtained from numeric time series by discretization.

If one is not interested in all future outcomes, but rather in predicting the occurrence of a certain symbol or pattern, supervised rule generation approaches (see Chapter 5.13) can be used.

5.7 Clustering

There are three different time series clustering problems. One can cluster a set of numeric time series based on some similarity measure and standard clustering algorithms. We call this

whole series clustering. The set of time series can be created from a single long time series by extracting short segments, we call this *sub-series clustering*. Finally, the time points of a time series can be clustered according to a combination of temporal proximity of the time points and the similarity of the corresponding values. This is similar to time series segmentation (see Chapter 5.8), we call it *time point clustering*.

5.7.1 Whole Series Clustering

There are numerous examples for clustering a set of univariate numeric time series. Das et al. calculate a complete similarity matrix with a robust LCSS metric for clustering 34 time series of telecommunication data (Das et al., 1997). Clustering of handwritten signatures is performed in (Chappelier and Grumbach, 1996) using the Euclidean distance and a small SOM with nine neurons. Debregeas and Hebrail also use very small maps (25 neurons)[2] in a software that visualizes clusters in electric load curves (Debregeas and Hebrail, 1998). (Smyth, 1997) uses a mixture of HMMs. The model is optimized with Expectation Maximization (EM) (Dempster et al., 1977) and cross-validation is used to determine the number of clusters. Hierarchical and partitional clustering is applied based on the precalculated similarity matrix. (Schmill et al., 1999; Oates et al., 2000) use the DTW distance and (van Wijk and van Selow, 1999) the Euclidean distance with hierarchical clustering. In (Oates et al., 2001) DTW is used to initialize HMM based clustering, because HMM training is sensitive to initial conditions and a HMM mixture cannot distinguish qualitatively different regimes per se. A fuzzy variant is used in (Alon et al., 2003), each time series is allowed to belong to each cluster to a certain degree. Several initialization methods are compared. In (Li and Biswas, 1999) Bayesian clustering with a mixture model of HMM is performed simultaneously selecting the number of clusters and the number of hidden states per model. In contrast to classification, where a discriminative border is modeled, this method is said to create interpretable characterizations for the clusters. This is demonstrated using plant growth data from mosquito control (Li and Biswas, 2002). A regression mixture model with an EM algorithm is used to cluster trajectories of hand movements (Gaffney and Smyth, 1999) and cyclone paths (Gaffney and Smyth, 2003). A distance measure for time series based on parameters derived from ARIMA models is proposed in (Kalpakis et al., 2001) and used with the partition around mediods clustering method. In (Sebastiani and Ramoni, 2001) an agglomerative clustering algorithm is described using the Kullback-Leibler divergence for AR models as a similarity measure. (Panuccio et al., 2002) calculate a pairwise similarity matrix using relative normalized likelihood distance of HMM. Xiong and Yeung use an EM clustering algorithm based on the likelihood distance of ARMA models together with Bayesian model selection (Xiong and Yeung, 2003). The method is applied to several different data sets in (Xiong and Yeung, 2004). The Euclidean distance of the ARMA parameters is used in (Bagnall and Janacek, 2004). In the experiments the method by Kalpakis et al. is not found to perform better than Euclidean distance if model selection with the Akaike Information Criterion (AIC, Hastie et al. (2001)) is used.

Clustering of time series from data streams is done in (Beringer and Hüllermeyer, 2003) using the Euclidean distance on the first few DFT coefficients and k-Means. Vlachos et al. developed an extension to the k-Means algorithm for time series based on DWT decomposition (Vlachos et al., 2003). Starting from coarse representations of the time series, more and more detail levels are used during the clustering process. In (Lin et al., 2003a) hierarchical and partitional clustering is used together with a symbolic representation of a numeric time series. In (Keogh et al., 2004b) the same representation is used together with a compression-based distance function. The clipped data representations is used for clustering in (Bagnall and Janacek, 2005).

[2]Note, that using small maps is essentially the same as running k-Means with k equal to the number or neurons (Ultsch, 1996a).

A SOM is combined with time series features for clustering in (Wang et al., 2004). An EM algorithms for the joint clustering and alignment of time series is described in (Gaffney and Smyth, 2004).

The method of (Keogh et al., 2004b) can also be used to cluster symbolic times series . Other approaches use the similarity functions for symbolic series (Chapter 5.3.2) in combination with clustering algorithms. A Gibbs sampling (e.g. Casella and George (1992)) and an EM method are developed for a mixture model of MCs in (Ridgeway, 1997). In (Sebastiani et al., 1999) a model-based distance for MCs is used to cluster symbolic sequences. The full Bayesian formulation including the number of clusters is given in (Ramoni et al., 2002). Law and Kwok formulate a version of Rival Penalized Competitive Learning (Xu et al., 1993) for HMM representation of symbolic sequences (Law and Kwok, 2000). Fischer and Zell use a fast approximation the Levenshtein string edit distance to speed up clustering of strings with a SOM (Fischer and Zell, 2000; Kohonen, 1999). Flanagan also uses a SOM with a feature-based distance function (Flanagan, 2003). In (Bicego et al., 2004) a mixture of the model-based and the feature-base approach is explored by training a HMM for a set of prototypical series and assigning each symbolic series the feature vector composed of the HMM output probabilities. A recent review of median strings as representatives for a set of strings is given in (Jiang et al., 2004).

5.7.2 Sub-Series Clustering

If there is just a single long time series, an obvious idea is to extract short series with a sliding window and cluster the resulting set of time series as described above. Care has to be taken though in the choice of the window width and the time delay between consecutive windows. The window width depends on the application, it could be some larger time unit, e.g., an hour for time series sampled in minutes. Using overlapping windows has been shown to produce meaningless results (Lin et al., 2003b). When using a time delay of one, almost all time points contribute equally to each position within the sliding window. Experiments with k-Means showed that clusters of sine waves are produced, the prototypes of which add up to the constant function. Using larger time delays for placing the windows does not really solve the problem as long as there is some overlap. Also, the less overlap, the more problematic the choice of the offsets becomes. If the delay is equal to the window width, the problem is essentially converted to whole series clustering (Lin et al., 2003b). We describe some historical approaches, an approach based on symbolic (sub-)sequences, and suggested solutions to make sub-series clustering meaningful.

(Das et al., 1998) is the most prominent example of meaningless sub-series clustering. They use a sliding window with unit delay to discretize time series and employ rule searching methods (see Chapter 5.13). Similarly in (Rosenstein and Cohen, 1998) robot sensor series are clustered using a delay of one. In (Fu et al., 2001b) sliding windows of different sizes are used in a multi-resolution manner and clustered with a small SOM (100 neurons), again the time delay is one. Ohsaki et al. use half the window width as the time delay.

In (Hebrail and Hugueney, 2000) non-overlapping windows are used. The width of the windows is chosen by investigating the periodical structure of the time series with the Fast Fourier Transform (FFT, Shatkay (1995b)). The offset of the first window was chosen such that the start is in a stable region. The resulting windows are clustered with a SOM of unreported size.

Denton presents a first approach to overcome the meaningless problem even when using overlapping subsequences by not forcing the algorithm to use all subsequences. A density-based clustering method with Gaussian kernels is used to find dense regions in the high dimensional space populated by the subsequences (Denton, 2004). Based on a random walk noise model, regions are identified that are significantly more dense than to be expected. The cluster prototypes represent shapes present in the original time series.

Hierarchical clustering of sub-sequences of a symbolic sequence is performed in (Ronkainen,

1998) based on an adapted edit distance. The sub-sequences are chosen as all symbols within an interval preceding a certain symbol of interest. The method is applied to telecommunication events and web log data.

5.7.3 Time Point Clustering

The clustering of time points is very similar to time series segmentation. One major difference is that with clustering not all points need to be assigned a cluster label. Many clustering algorithms allow for points to be labeled as noise (Berkhin, 2002), while segmentation algorithms usually produce a partition of the time points. Another important difference is that clustering usually produces recurrent labels (Gionis and Mannila, 2003), i.e., the points of separated segments can belong to the same cluster. In contrast, segmentation produces unrelated intervals. The time point clustering problem is described formally in (Gionis and Mannila, 2003): The problem of finding h hidden sources occurring k times within a sequence is called (k, h)-segmentation. For $k > h$ this is time point clustering, for $k = h$ we call it segmentation.

Any clustering algorithm can be used to cluster the values of a univariate time series or the vectors of a multivariate time series ignoring the temporal order. If the time points of each cluster appear as consecutive stretches instead of being randomly permutated, this can be seen as a validation of the found clusters. Emergent SOM (ESOM, Ultsch (1993); Ultsch and Mörchen (2005)) clustering of the time points of a multivariate time series was used in (Guimarães and Ultsch, 1999) and (Mörchen et al., 2006). In (Flexer and Bauer, 1999) the vectors of a multivariate time series were clustered with k-Means to obtain a set of codebook vectors for further processing. Poliker and Geva use fuzzy clustering of the data space and project the membership functions on the time axis to model time series with continuous change in regime (Poliker and Geva, 1998). The membership functions are averaged over short periods of time.

In order to incorporate the time aspect more systematically, the Viterbi (Rabiner, 1989) algorithm can be used to obtain a state sequence from a HMM trained on the time series. Gaussian output distributions can be used for a piecewise constant model or regression curves for more complex shapes (Ge and Smyth, 2000).

Gionis and Mannila show that the (k, h)-segmentation problem is NP-hard in the general case (Gionis and Mannila, 2003). They propose two simple algorithms and a combined iterative algorithm inspired by EM. Theoretical bounds for the approximation capabilities of the algorithms are given.

5.8 Segmentation

A good review on the most common segmentation methods for univariate time series in the context of piecewise linear representation is given in (Keogh et al., 2004a). Three basic approaches are distinguished: *"Sliding Windows: A segment is grown until it exceeds some error bound"*, *"Top-Down: The time series is recursively partitioned until some stopping criteria is met"*, and *"Bottom-Up: Starting from the finest possible approximation, segments are merged until some stopping criteria is met"*. These algorithms are commonly used in favor of the optimal but computationally expensive solution (Bellman, 1961).

The Sliding Window approach is generally problematic because the performance heavily depends on the data set used (Shatkay, 1995a; Keogh et al., 2004a). Choosing the right window size can be difficult depending on the scale of the interesting shapes in the time series and the amount of noise present. It also showed poor performance in an evaluation with many real life data sets (Keogh et al., 2004a). The Top-Down approach (Shatkay, 1995a; Li et al., 1998; Park et al., 1999; Lavrenko et al., 2000) has time complexity $O(n^2)$ where n is the size of the time series. It is usually qualitatively outperformed by Bottom-Up (e.g. Keogh and Smyth (1997);

Keogh and Pazzani (1998); Hunter and McIntosh (1999); Last et al. (2001)), which further only scales linearly with the size of the data set. In contrast to sliding windows, the latter two approaches can only be run in batch mode, however. Based on these observations the Sliding Window and Bottom-Up approach was developed to combine the incremental nature of Sliding-Window with the more global view of Bottom-Up (Keogh et al., 2004a).

Himberg et al. present fast greedy algorithms to improve a segmentation found by other methods or simply by initial uniform placing of cut points. Local and global approaches are used to reconsider the placement of cut points, moving them into local minima of the cost function (Himberg et al., 2001a). The global approached performed best on data from context recognition.

All methods mentioned so far need to be given either the number of segments or an error threshold a priori. Both can be difficult to select, especially with very long time series. A statistical method for choosing the number of segments based on permutation tests is described in (Vasko and Toivonen, 2002). The idea is to look at the relative reduction of the cost function when going from $k-1$ to k segments and compare this to permutated versions of the time series. The criterion outperformed other common model selection techniques on artificial data and a time series from archeology. It can be combined with any segmentation method. In (Salvador et al., 2004) scree plots are used to automatically select the number of segments.

In order to decide which shape features of a time series are important and which are caused by noise a multi-scale analysis can be used (Lindeberg, 1993; Bakshi and Stephanopoulos, 1995; Mallat, 1999; Höppner, 2002c, 2003). Bakshi and Stephanopoulos use a first order bi-cubic spline DWT to detect the inflexion points of a time series at several resolutions. This creates a so-called interval tree of scales, a hierarchical tiling of the time frequency plane. The importance of an interval is measured using a heuristic based on the lifetime of an interval in the scale-space (Witkin, 1983). For a detailed discussion of the advantages of scale space filtering over other segmentation methods see Chapter 2.2 of (Höppner, 2003). Sripada et al. especially discuss the interpretation aspect of segmented time series. They observe, that different error thresholds can be necessary depending on the range of the variable. The stopping criterion is also very user dependent (Sripada et al., 2002).

Usually the time series is modeled with a low order polynomial on each segment. Sometimes more complex shapes are sought, e.g., domes or ramps for data from gas turbines (Sripada et al., 2002) or complex patterns in stock prices (Fu et al., 2001a). The latter approach uses evolutionary algorithms to segment a time series with the goal of finding such arbitrarily shaped patterns.

Few approaches consider multivariate time series. The problem can be formulated as clustering with the constraint that all time points of a cluster need to be continuous in time. This is used in (Mäntyjärvi et al., 2001) to segment sensor data from mobile phones. A fuzzy clustering algorithm is used in (Abonyi et al., 2004) to segment a multivariate series from an industrial process. The cost function is modified to favor clusters contiguous in time. It is noted that choosing the number of segments, is problematic. In (Gionis et al., 2004) the problem is extended to find a partition of the dimensions and segment each subset individually. Compared to segmentations of the complete multivariate time series with the same number of parameters, the so called clustered segmentations found with a k-Means like algorithms are generally better w.r.t. the variance on the segments. For a real life example a meaningful grouping of the dimensions was found.

Time series segmentation is related to change point detection. Statistical change point methods usually depend on an assumed model of a stochastic process producing the time series. A more simple method based the model of linear functions per segment is described in (Guralnik and Srivastava, 1999). In (Oliver et al., 1998) a time series is assumed to be produced by a Gaussian distribution per segment. The number and location of the segments and the dis-

tributions are estimated simultaneously. Approximations for large data sets are described in (Fitzgibbon et al., 2002).

By changing the underlying probability distributions, these methods can easily be applied to symbolic time series as well. But the problem of specifying a model remains. An entropy based approach for symbolic data without model assumptions is described in (Cohen and Adams, 2001). The task is to separate a series of symbols into meaningful segments. The intuition is that within such a segment the series is more predictable than at the borders. The approach is successfully applied to discover words in a text with spaces and punctuation removed.

5.9 Classification

There are two types of classification in time series data mining. The first is *time series classification* similar to whole series clustering. Given a set of time series with a single label for each, the task is to train a classifier and label new time series. In *time point classification* problems there are labels for each time point. A classifier is trained using the values and labels from the time points of the training set. Given a new time series, the task is to label all time points. This is called sequential supervised learning in (Dietterich, 2002) and related to prediction (see 5.6).

5.9.1 Time Series Classification

An early approach to time series classification is presented in (Bakshi and Stephanopoulos, 1994). The time series are represented by shape primitives derived from a multi scale analysis. This results in a combination of qualitative and quantitative features. Decision trees are induced in an iterative process going from coarse to more detailed time series representations as necessary. The method is applied to the classification of chemical process trends. Only few records were used, but the results could be interpreted meaningfully.

Many approaches are basically a combination of temporal feature extraction and a conventional classification scheme. A piecewise polynomial representation is used with Bayesian classification in (Manganaris, 1995). Each class is represented by a set of models. A piecewise linear representation including weights per segment is proposed in (Keogh and Pazzani, 1998). The weights can be used in relevance feedback, e.g., for classification. The same representation is used in (Geurts, 2001) where classification rules are induced with decision trees.

(Deng et al., 1997) use ARMA models and perform nearest neighbor classification using the Euclidean distance of the model parameters. The order of the models is estimated globally using the AIC criterion. HMM representations are used in (Zhong and Ghosh, 2002). Neural networks are trained on statistical features in (Nanopoulos et al., 2001). In (Povinelli et al., 2004) a phase space representation is combined with Bayesian classification based on Gaussian mixture models.

The FeatureMine method (Lesh et al., 1998) searches sequential patterns in symbolic sequences (see Chapter 4.2) that are frequent, distinctive, and non-redundant. The boolean features indicate the presence of a pattern in a symbolic time series and are used to train a static classifier. In (Oates, 1999) sub-series of multivariate numerical time series that are distinctive for a given class are searched.

Wavelet features are used with recurrent neural networks in (Roverso, 2000) to classify short multivariate time series. The wavelet coefficients with the lowest global reconstruction error are used in (Mörchen, 2003) with rule generation approaches to obtain classification rules. In (Zhang et al., 2004) the Haar wavelet is used to determine the appropriate scale using the entropy of the detail coefficients. The extracted features are tested with nearest neighbor classification. The classification of symbolic time series with wavelet features is described in (Aggarwal, 2002).

The TClass method for time series classification with comprehensible descriptions for multivariate time series has been proposed in (Kadous, 1999) and was later extended (Kadous, 2003; Kadous and Sammut, 2004, 2005). The first step is the extraction of global features, like the mean of the time series. The temporal structure is described, e.g., by trends and local minima and maxima. The second step is a clustering of the features. The cluster centroids are used as inputs for the training conventional classifiers, e.g., C4.5. The classification rules produced by C4.5 were post processed to convert rules about the cluster centroids back to the original features. The method is applied to time series describing the Australian sign language and the class descriptions were said to correspond to the official sign definition. A tradeoff between more comprehensible descriptions using simple classifiers and accuracy using bagging or boosting (e.g. Quinlan (1996)) is observed (Kadous and Sammut, 2005).

Another method targeted at comprehensible descriptions of time series is described in (Rodriguez et al., 2000; Rodriguez and Alonso, 2000; Alonso and Rodriguez, 2000). The time series are described with interval and distance based predicates and first order logic. This creates representations like *mostly increasing on interval [5, 12]*. Classification rules are created with inductive logic programming (Lavrac and Dzeroski, 1994). The accuracy was optimized using boosting (Schapire, 1999) of the base literals. The results were competitive on several benchmark data sets and the Australian sign language data set. The ensemble classifiers created with boosting sacrifice interpretability of the logical predicates, however. The method was recently extended to variable length time series and to provide early classification based on a prefix of the time series (Alonso and Rodriguez, 2004).

Most of the above approaches have a separated (temporal) feature extraction and apply a static learning scheme to the result. Recently more tighter integration of temporal structure into the learning algorithms have been proposed. Yamada and Suzuki use a decision tree with a modified split test (Yamada and Suzuki, 2003). No features are extracted, whole time series are stored in the tree nodes compared with the DTW distance. They note that the complete time series can be more comprehensible to experts than extracted features. Ratanamahatana and Keogh also use DTW distances with nearest neighbor classification. The class information on the training set is used to learn class wise constraints of the DTW warping path. This results in excellent accuracies on several data sets (Ratanamahatana and Keogh, 2004b).

5.9.2 Time Point Classification

Time point classification methods label time points. Not all time points need to be labeled if only certain interesting events need to be detected. Several methods for time point classification are reviewed in (Dietterich, 2002). Simple methods use vectors of values extracted from the time series with a sliding window and apply conventional learners. The recurrent sliding window approach also uses the labels of already classified points. This includes recurrent decision trees and recurrent neural networks. Further, the application of HMM variants, Conditional Random Fields, and Graph Transfer Networks to time point classification are described (Dietterich, 2002).

Cotofrei and Stoffel developed a point classification method based on first order logic (Cotofrei and Stoffel, 2002c,a, 2005). Each time series is discretized using equal frequency histograms on the first order differences or segmentation to describe local shapes. This creates a symbolic feature vector for each time point. Classification trees are trained with these feature vectors using a sliding window to incorporate temporal order. The variation of parameters like the number of symbols per time series or the width of the sliding window creates multiple training sets. The rules for a class derived from the corresponding trees are combined with logical operators. The method is extended with higher order mining in (Cotofrei and Stoffel, 2002b, 2005). A two step generalization is performed on the extracted classification rules. Rules with

a confidence value below a statistically derived threshold are deleted. The remaining rules are merged and further pruned using the MDL principle.

5.10 Motif Discovery

The discovery of time series motifs was motivated by the observation that sub-series clustering with overlapping sliding windows produces meaningless results (Lin et al., 2003b). The idea of motifs was transferred from symbolic gene time series in bioinformatics to numerical time series. Motifs are defined in (Lin et al., 2002; Patel et al., 2002) as typical non-overlapping subsequences.

An extension to motifs with gaps is proposed in (Chiu et al., 2003) inspired by the usage of random projection for the discovery of DNA motifs (Buhler and Tompa, 2001). The time series are represented with PAA and fixed length sub-series are extracted. Several projections using a random subset of symbol positions of the sub-series are created and collisions of similar patterns are counted. Two similar patterns will have more collisions than different patterns.

The SAX representation is combined in (Lin et al., 2004) with a tree visualization of sub-series to identify Motifs manually. (Tanaka and Uehara, 2003) extend Motif discovery to multivariate time series using PCA and MDL. In (Liu et al., 2005) motifs are found with a density based method.

5.11 Anomaly Detection

The detection of anomalies, also commonly called novelty detection, has applications like fault detection in machine monitoring or intrusion detection in computer networks. The events of interest are usually rare. A good discussion on the difficulties of mining rare events is given in (Weiss, 2004). Common quality measures like accuracy or information gain are not well suited, because of the extreme difference in the size of the classes to be predicted. Many data mining algorithms are found to have the wrong induction bias, e.g., greedy search. Weiss proposes to use precision and recall as quality scores and non greedy search methods like genetic algorithms.

One approach to detect anomalies is to create a model of the normal state of a time series. Incoming data that does not fit the model well is characterized as abnormal. This is done in (Ypma and Duin, 1997) using SOM models of the normal data. If the projection error of new data on the SOM is larger than a threshold it is classified as an anomaly. A framework for (online) novelty detection is defined in (Ma and Perkins, 2003) and implemented based on Support Vector Regression (SVR) (Müller et al., 1997).

The principle of negative selection from immunology is used in (Dasgupta and Forrest, 1996) for anomaly detection. A set of detectors that *fail* to detect the training time series is generated. These detectors are later used for monitoring. If incoming data triggers a detector, it is classified as an anomaly. The negative selection is extended to numerical time series in (Gonzalez and Dasgupta, 2002). Supervised Multi Layer Perceptron (MLP) neural networks and unsupervised SOM are compared. A genetic algorithm is used in (Gomez et al., 2003) to generate fuzzy rules describing non-self. In (Oliveira et al., 2004) anomalous examples are generated to train MLP and RBF neural networks.

(Eskin, 2000) does not require supervised training, but assumes anomalies to be rare. A mixture model is trained on all incoming data. Anomalies are detected by a statistical test against this model. The approach is applied to symbolic time series from computer monitoring (Eskin, 2000).

A wavelet decomposition of time series is used in (Shahabi et al., 2000) to support queries for surprises at different resolution levels. The TARZAN algorithm (Keogh et al., 2002) is based

on a MC model of a discretized version of the time series. Suffix trees are used to store fixed length sub series along with the occurrence probabilities. If incoming data does not fit the model trained with normal data, it is labeled as a surprise. Experimental results show superior performance over the immunology and wavelet methods.

A state based approach is taken in (Salvador et al., 2004) that consist of several processing steps. First, the time series are segmented using time point clustering. The clusters are assumed to be states of the process generating the time series. The RIPPER (Cohen, 1995) algorithm is used to generate characterization rules for the clusters. A finite state automaton is constructed with one state per cluster plus an abnormal error state. The state transitions are determined by the cluster rules. If no cluster rule is triggered by incoming data, the error state is activated. If the error state occurs too many times consecutively or totally, an anomaly is reported.

Time series bitmaps are used in (Wei et al., 2005) for a simple unsupervised anomaly detection algorithm. The Euclidean distance between the bitmap representations of two windows leading up to and following a time point is used as an anomaly score.

5.12 Retrieval by Content

The task of retrieving a set of time series from a database that are similar to a given example is a common problem in the database community. It is often combined with an indexing strategy to speed up the retrieval. In early stages time series data mining was almost exclusively occupied with this topic starting with the seminal work of Agrawal et al.. It remains an active area of research with recent developments like indexing based on the DTW distance function (Keogh and Ratanamahatana, 2005; Zhu and Shasha, 2003; Sakurai et al., 2005). Many publications are composed of a particular time series representation (see Chapter 5.4), a similarity measure invariant under some transformations (see Chapter 5.3), and an indexing method. A recent survey covering retrieval and indexing of time series can be found in (Hetland, 2004).

In (Agrawal et al., 1993a) the time series were represented by the first few coefficients of the DFT to reduce the dimensionality by a large factor. The feature representation was queried with a R^*-Tree (Beckmann et al., 1990) index. Since the Euclidean distance of the DFT coefficients lower bounds the Euclidean distance of the original time series, there are no false dismissals. False hits were removed in a post-processing step using the original data. Many papers extended this approach, e.g., by changing the feature representation. In (Rafiei and Mendelzon, 1998) the first and the last k DFT coefficients were used exploiting the conjugate property of the DFT on real valued signals. The first few coefficients of the DWT were used in (Chan and Fu, 1999). A comparison of DFT and Haar DWT (Wu et al., 2000) showed, that they are comparable in energy preservation, while DWT is faster to calculate and offers a multi-resolution decomposition. In addition there are approaches that combine DWT with time warping (Chan et al., 2003). In general, DFT will be better for time series with a strong periodic structure, while DWT will be better for time series with important local shapes and singularities (Mörchen, 2003). Popivanov and Miller pursued the suggestion by Chan and Fu, that wavelet bases other than Haar, might be better for certain data sets. They showed that the contraction property needed for indexing holds for all (bi-)orthogonal wavelet transforms, making a large family of transforms available for the feature extraction (Popivanov and Miller, 2002). In particular the Daubechies wavelet family (Daubechies, 1992) was shown to have good energy preservation on several real life and synthetic data sets. Recently, random projections (Indyk et al., 2000) and Chebychev polynomials (Cai and Ng, 2004) have been used as feature representations for time series retrieval.

A simple symbolic representation based on a shape alphabet with fixed resolution was proposed in (Agrawal et al., 1995b). An index structure for queries supporting regular expression is described. (Qu et al., 1998) present an extension to user specified scales. The piecewise

linear approximation (PLA) is used in (Shatkay and Zdonik, 1996), similar representations are proposed in (Perng et al., 2000; Pratt and Fink, 2002; Fink and Pratt, 2004). An extension of PLA to deformations and gaps is described in (Keogh and Smyth, 1997). The advantage of more flexible distance functions is shared by the piecewise constant PAA and APCA representations (Keogh et al., 2001b; Yi and Faloutsos, 2000; Keogh et al., 2001a). APCA is also shown to have more pruning power than DWT or DFT features (Keogh et al., 2001a). The most recent representation in this line of work is SAX, the only symbolic representation with a distance function lower bounding the Euclidean distance (Lin et al., 2003a).

Other important extensions to the general approach of (Agrawal et al., 1993a) include the handling of scaling and gaps (Agrawal et al., 1995a; Vlachos et al., 2002), noise (Vlachos et al., 2004a), formalizing query constraints and incorporating them into the indexing procedure (Goldin and Kanellakis, 1995; Rafiei and Mendelzon, 1997; Goldin et al., 2003) and subsequence matching (Faloutsos et al., 1994; Moon et al., 2001; Kahveci and Singh, 2001). The most interesting transformation to handle is time warping. Since the DTW distance does not satisfy the triangle inequality (Yi et al., 1998) it is difficult to index. An early approach achieved a significant speedup but allowed false dismissals (Yi et al., 1998). A lower bounding distance for DTW was claimed in (Park et al., 1999), but retracted in (Park et al., 2000). A segment based calculation was then used to speedup the processing. Lower bounding distances without false dismissals for DTW were then proposed in (Kim et al., 2001) and (Keogh, 2002). Extensive experiments in (Keogh and Ratanamahatana, 2005) show the lower bounding distance of Keogh and Ratanamahatana to be superior to that of Kim et al. and the approximation of Yi et al.. The method was further improved in (Zhu and Shasha, 2003). The lower bound relies on commonly used constraints of the warping path. The Fast Time Warping (FTW, Sakurai et al. (2005)) method can be used without constraints and is shown to outperform the method of (Keogh and Ratanamahatana, 2005) on some data sets. In (Fu et al., 2005) a lower bound for DTW combined with uniform scaling is given.

Other recent trends in time series retrieval include multivariate time series and specialized index structures. Extensions of the DTW indexing to multivariate time series are reported in (Rath and Manmatha, 2003). The indexing method for Euclidean distance from (Agrawal et al., 1993a) is applied to multivariate time series in (Lee et al., 2000). The method proposed in (Cai and Ng, 2004) can trivially handle multiple dimensions. (Kahveci et al., 2002) also covers the multivariate case handling scaling and shifting in a specialized indexing structure called ConeSlice. The Skyline index (Li et al., 2004) for time series can be combined with several time series representations and introduces bounding regions in the time series space instead of typically used minimum bounding rectangles in the feature space (Faloutsos et al., 1994). The DDR index (An et al., 2005) uses a grid based representation to overcome the problems of spatial indexing methods with high dimensionalities. In (Cheng et al., 2005) an indexing method for symbolic sequences based on sequential patterns is described.

5.13 Rule Discovery

Rule discovery is a crucial task for knowledge discovery, because the output is often directly aimed at humans providing insights into the data. Even though in some cases the presentation of the original time series to an expert might be useful (Yamada and Suzuki, 2003), generally knowledge discovery requires an abstraction mechanism that creates symbolic rules close to natural language or easily convertible to natural language.

Rule discovery from time series can be done supervised and unsupervised. In supervised approaches something about the rule to be found is known a priori, at least for part of the data. This mainly involves prediction of known events. Unsupervised approaches search for typical temporal patterns that can be described by rules. Similar to association rules a rule pattern

is sometimes transformed to an implication rule with premise and conclusion. We will only describe supervised approaches in this chapter. The unsupervised discovery of rules from time series is deferred to Chapter 6.

5.13.1 Supervised Rule Discovery

The prediction of known rare symbols in symbolic time series is performed by the TimeWeaver method (Weiss and Hirsh, 1998; Weiss, 1999; Weiss and Hirsh, 2000). The prediction rules are expressed in a simple rule language with wild cards covering the temporal concepts of order and concurrency.

(Sun et al., 2003) searched for so-called event oriented patterns that frequently occur in a window prior to the target symbol. The patterns are based on Episodes (Mannila et al. (1997), Chapter 6) and cover the temporal concepts of order and concurrency. The search for negative patterns, i.e., patterns that rarely occur before the target symbol is described in (Sun et al., 2004b). If a pattern occurs far less than expected in a window before the symbol of interested this can just as well be used as a predictor as frequent patterns. Patterns frequently occurring in windows not preceding a target symbol are searched in (Sun et al., 2004a). (Sun et al., 2005) describes an extension to mine the minimum time window for the patterns to be still considered interesting.

In (Bettini et al., 1998) temporal patterns with a pre-specified structure are searched. The patterns consist of symbolic variables connected with temporal constraints in a finite automaton. The temporal constraints are defined w.r.t. user defined granularities, e.g., business days. The search for pattern instances includes instantiation of the variables and counting of the occurrences. The first symbol in the patterns needs to be specified a priori. The authors suggest to interactively mine simple patterns first and move towards more complex patterns based on the findings. A similar approach is described in (Li et al., 2001) supporting multiple granularities and comparing several mining algorithms.

Interval sequences obtained from videos are described with rules in (Fern et al., 2002). The rules consist of logic formulas based on the AMA language (see Chapter 3.2.3) and are learned using inductive logic programming. Based on positive examples only, the least general covering formulas of each example are combined to obtain the least general generalization of all examples. The rules are comparable in accuracy to a hand coded set of rules for the same application, but not as comprehensible.

A complex system for temporal rules based on heterogeneous data including time points and time intervals is described in (Shahar, 1997). Higher level patterns are obtained from lower level information using temporal reasoning techniques and systematically incorporating domain knowledge. We therefore consider it a supervised approach. The usefulness is demonstrated, e.g., for clinical domains (Y. and M.A., 1996).

5.14 Other Approaches

The following approaches did not clearly fit in any of the above categories for time series data mining tasks. In (Oates, 1999) numerical patterns are searched in time series to distinguish instances likely to lead to a pre-specified event. In (Oates, 2002) multivariate patterns are described by a temporal sequence of normal probability distributions. A supervised EM based algorithm and an unsupervised dynamic programming based learning method are described.

The framework proposed by Povinelli uses sub-series patterns to predict a given event, e.g., the sudden rise of a stock price (Povinelli, 2001). A genetic algorithm is used to find patterns based on a phase space representation (Povinelli and Feng, 2003).

In (Savnik et al., 2000; Lausen et al., 2000) a method for finding the common characteristics of a set of time series is described. For each predefined interval, a model is constructed using classification with, e.g., C4.5. The model can be used to align undated time series with the set of training time series.

5.15 Discussion

After describing the various different TSDM tasks, we recall the main problems and comment on the state of the art.

A fundamental concept is that of time series similarity, strongly interconnected with time series representation. Many common tasks, e.g., retrieval, directly depend on a good specification of the similarity measure and an accompanying representation supporting efficient processing. Many different similarity measures and representations have been proposed (Keogh and Kasetty, 2002). For certain applications there is surely still room for improvement. Especially feature and model based representations are highly problem dependent. For the discovery of understandable knowledge it is important to choose a representation that can be interpreted by the expert.

Retrieval, being the first highly researched topic in TSDM, is in a rather mature state. Nevertheless, there are recent publications specializing on time series (Li et al., 2004; An et al., 2005) instead of plugging together standard methods with feature based time series representations. The understandability is usually of minor importance when retrieving approximate matches.

Clustering relies strongly on a good choice of the similarity measure. There are many publications on whole series clustering, with a recent trend towards methods that better integrate the temporal aspects (Vlachos et al., 2003). Whether sub series clustering can produce meaningful results is still being disputed (Lin et al., 2003b; Struzik, 2003; Denton, 2004). There seems to be hope in not using all possible windows. Time point clustering is very similar to segmentation, the main difference being that clusters are usually recurring. This is an important difference for rule discovery, because often frequently repeating patterns are searched. For time series segmentation there are several established algorithms (Keogh et al., 2004a). First solutions have been proposed to select the number of segments (Vasko and Toivonen, 2002; Salvador et al., 2004).

Motif discovery is a new research topic with few publications. The set of most frequent motifs can be used to summarize a time series or generate rules. Motifs can further solve the sub-series clustering problem because they are non overlapping by definition.

Time series description can sometimes be done directly based on the representation, e.g., using feature based or model based representations. For shape based problems the work of (Sripada et al., 2003b) offers an end-user compatible natural language representation that has high potential for successful knowledge discovery.

Classification is dominated by approaches combining classifiers for static data with preprocessing techniques transforming the time series into a suitable representation (Kadous, 1999; Nanopoulos et al., 2001). Few methods are explicitly designed for time series (Rodriguez et al., 2000; Yamada and Suzuki, 2003; Ratanamahatana and Keogh, 2004b). It is surprising that the early approach of (Bakshi and Stephanopoulos, 1994) offering a sound methodology for interpretable representation and classification of time series does not seem to be widely known. Also, there is a lack of studies comparing different approaches in accuracy and interpretability.

Time point classification is far less common, a promising method w.r.t interpretability is (Cotofrei and Stoffel, 2002c). There is a strong relation to prediction, where only future points are of interest. In anomaly detection the focus is only on the time points of where anomalies occur. Here, several methods have been proposed but are hardly ever compared to each other (Keogh et al., 2002). Supervised rule discovery can be seen as a general form of time point classification, if it is concerned with the time points. But rule discovery in general has usually the aim of producing understandable results, not necessarily the most accurate predictions.

In summary, a lot of work has been done in TSDM. Starting with approaches in retrieval a variety of different tasks has been recognized and solutions have been proposed. As already mentioned in (Keogh and Kasetty, 2002), there is still the need for more comparative studies that evaluate different methods on several data sets with different characteristics. It seems that only recently researchers have gone beyond simply combining conventional methods with time series feature extraction and start to integrate the temporal characteristics of the data more tightly into the mining algorithms.

Unfortunately, the results of many time series data mining methods can not be directly used for knowledge discovery, because they are hard to interpret. A neural network classifier based on a statistical feature vector (Nanopoulos et al., 2001) makes it hard to understand the decision for a particular class. Even methods claiming to achieve high interpretability, sometimes fail to achieve this goal. The classification rules generated in (Kadous, 1999) can only be interpreted meaningfully, if the intermediate feature based representation of the time series is manually designed to be descriptive and understandable. (Rodriguez et al., 2000) use logical predicates describing aspects of the time series, but the classification is done using boosting. To help understand a decision for a class, one would have to look at all weights per predicate and class. For anomaly detection the approach in (Salvador et al., 2004) offers interpretation of the normal states with rules, but not for the anomalous states.

The generation of natural language descriptions for complete time series (Sripada et al., 2003b) is an important step on the way to extract useful and understandable knowledge from time series. But the representation always needs to be adjusted to the users experiences and capabilities (Sripada et al., 2003b). Often only certain parts of a time series are relevant or interesting. The unsupervised discovery of symbolic rules from time series offers understandable description of local phenomena, existing approaches will be discussed next.

Chapter 6

Unsupervised Mining of Temporal Rules from Time Series

In this chapter we describe existing approaches to unsupervised rule discovery in time series. The rule discovery methods are categorized by the data model they use for rule generation (see Chapter 3.1) and the temporal concepts the rules can express (see Chapter 3.3). Many of the following approaches have been developed with a certain application in mind and use only few temporal concepts. There are two types of data models used for rule mining, namely time point and time interval data types, almost exclusively of the symbolic kind.

Numerical time series are often converted to symbolic time series or interval time series in a preprocessing and feature extraction stage (Das et al., 1998; Guimarães and Ultsch, 1999; Villafane et al., 1999; Last et al., 2001; Höppner, 2001; Harms et al., 2002; Himberg et al., 2003; Saetrom and Hetland, 2003). Various methods for this conversion were described in Chapter 5 and are reviewed briefly.

For univariate numeric time series the conversion to symbolic form can be performed based on single values or local shape of the time series. Value-based discretization is most commonly used. The resulting symbols can be mapped to linguistic descriptions like *high* or *low*. Shape-based discretizations describe the development of a time series on an interval. The intervals can be overlapping or consecutive and can be of equal or varying length. Variable length intervals can be obtained by time series segmentation (Chapter 5.8). Fitting linear or quadratic functions and discretizing the values of slope and curvature corresponds to descriptions like *increasing* or *convexly decreasing*. Alternatively, the sub-series can be clustered (Chapter 5.7.1) to create a codebook of prototypes. Meaningful descriptions for the symbols corresponding to the cluster centers have to be found separately, e.g., with (Sripada et al., 2003b).

For multivariate numeric time series time point clustering (Chapter 5.7.3) can be used to obtain a symbolic time series. To enable interpretation of the symbols, rules characterizing the clusters can be generated (Ultsch, 1999; Witten and Frank, 1999).

The Viterbi algorithm can be used to obtain a symbolic time series from a Hidden Markov Model (HMM, Rabiner (1989)) trained on a univariate or multivariate numeric time series. HMM models are harder to interpret than range partitioning for univariate data or clustering and rule generation for multivariate data, however.

In the following examples for rules obtained from symbolic time point or time interval data, t will be used to indicate a time point, δ for time lags, and upper case variables for rule parts, i.e., single symbols, intervals, or more complex patterns. All example rules usually hold with a certain confidence or probability.

6.1 Time Point Rules

The most commonly searched temporal concept in univariate symbolic time series is order, i.e., characteristic short sub series of symbols occurring sequentially. This is a typical problem of string matching and computational biology, (e.g. Gusfield (1997)). Many approaches in the analysis of gene data are concerned with multiple alignment and patterns present in several symbolic time series, however. The discovery of typical patterns in a single symbolic time series can be done by constructing a suffix tree or a variant thereof. The method described by Vilo supports patterns with wild cards and group characters. A pattern trie with statistics in the tree nodes is built to efficiently find the most frequent patterns in a long symbolic time series (Vilo, 1998, 2002). The higher the minimum frequency is set, the faster the algorithm runs. The efficient calculation of several interestingness measures for subsequences is described in (Apostolico et al., 2000). Over- and underrepresented sequences w.r.t. a binomial or Markov background model can be found. The visualization of the interestingness can be done with suffix trees and different font sizes and colors for the nodes (Apostolico et al., 2003). In (Jiang and Hamilton, 2003) several traversal strategies of a suffix trie are compared for efficiency. In (Cohen et al., 2001) a tree is built from a multivariate binary time series including mismatch costs of the binary vectors based on the Hamming distance.

Symbolic time series and sequences can be viewed as a special case of itemset sequences where each itemset is of cardinality one. The methods for mining sequential patterns in itemset data (see Chapter 4.2) can also be used to discover the concept of order. A long sequence needs to be split into many short sequences, however.

In (Cohen and Adams, 2001) patterns in symbolic time series are distinguished from so-called Episodes. Note that this is different from the Episodes in the line of research following (Mannila et al., 1995) described below. While a pattern is an arbitrary sequence of symbols, an Episode is defined by (Cohen and Adams, 2001) as a pattern with a semantic meaning to the expert. To avoid confusion we will therefore call them Semantic Patterns in the further discussion. The search for the borders of such a Semantic Pattern is based on the entropy of the probability distributions of symbols following a subsequence. Within a Semantic Pattern the entropy for the next position starts high and decreases because the following symbols become more predictable. At the end of a Semantic Pattern this entropy should peak. A simple algorithm for finding these boundaries using several window widths and a scoring theme is described. The resulting patterns can be arbitrarily long but do not contain group characters or wild cards.

In (Das et al., 1998) the concept or order is mined in univariate symbolic time series with simple rules expressing the relation of two symbols only. Sequential occurrences of two symbols with a maximum temporal distance are searched. A rule can be read as: *if A occurs at time t, then B occurs before* $t + \delta$. The rules are evaluated with the J-measure (Smyth and Goodman, 1991). The method is said to be easily applicable to multivariate symbolic time series by considering single symbols from all dimensions. In the same way it could also be applied to symbolic sequences. An extension to more than two symbols in a rule is discussed as more difficult and computationally expensive.

Oates et al. describe a similar approach to discover the concept of order in multivariate symbolic time series, the Multi Stream Dependency Detection (MSDD) (Oates et al., 1996, 1997). Each rule element contains a symbol or a wild card for each dimension of the time series. A rule consists of two such elements and reads: *if A occurs at time t, then B occurs at* $t + \delta$. Thus for d dimensions a rule consists of at least two and at most $2d$ symbols. Note, that the temporal constraint is more strict than in (Das et al., 1998). The related method Multi Event Dependency Detection (MEDD) (Oates et al., 1997, 1998) discovers rules representing the concept of concurrency in symbolic time sequences. The same basic patterns are used but the rules express: *if A occurs at time t, B occurs between* $t - \frac{\delta}{2}$ *and* $t + \frac{\delta}{2}$. In both methods, the

rules are defined as interesting if the probability is far from expectation. This is measured with contingency tables of the (co-)occurrences of both rule parts. The search space is systematically searched in a general to specific manner. Possible performance problems can be overcome by parallel implementations (Oates et al., 1996). MEDD can also be applied to univariate or multivariate symbolic time series, mining the concept of concurrency instead of order.

The Episode patterns of (Mannila et al., 1995, 1997) describe a partial order of elements in a symbolic time sequences and represent the concepts of order and concurrency. Serial Episodes express order, because all symbols must occur sequentially. The length of the pattern is restricted by a maximum temporal distance between the first and the last symbol. In parallel Episodes the symbols of a pattern can occur in any order within the window, representing the concept of concurrency. Partially ordered Episodes mix these two temporal concepts. An Apriori style algorithm is used to build longer candidate patterns from frequent short patterns. The type of patterns and the width of a sliding window in which to search needs to be specified by the user. An Episode rule is built from a frequent Episode similar to association rules by looking at sub patterns. Let the following simple serial Episode be frequent: *symbol A followed by B followed by C within at most δ time points.* A possible Episode rule generated from this reads: *if A occurs at time t followed by B, then C will occur before $t + \delta$.* Given the frequent parallel Episode: *A,B,C occur within a window of length δ*, a parallel Episodes rule could be: *if A occurs with B in a window of length δ then so does C*. Note that the premise of Episode rules doesn't necessarily occur temporally before the conclusion, even for serial Episodes. Thus the usability of such rules for prediction is not always given. Typical applications for Episode rules include analysis of telecommunication data or web server logs, where the interest is mostly in several events happening temporally close but in no particular order.

Determining support and confidence of Episode rules requires counting the occurrences of Episode patterns in windows. The WINEPI (Mannila et al., 1995) method uses the relative frequency, i.e., the number of windows with the pattern divided by the total number of windows. The MINEPI (Mannila and Toivonen, 1996) method uses the concept of minimal occurrences, i.e., a window around the pattern that does not contain sub windows with the same pattern. In (Baixeries et al., 2001; Casas-Garriga, 2003) these approaches are criticized for the need of a fixed maximum window length. Instead they constrain the maximum distance between successive symbols in a pattern. This way an Episode pattern can be arbitrarily long. The frequency counting is performed with respect to the pattern length. Méger and Rigotti show that the method of (Casas-Garriga, 2003) is incomplete and proposed the WinMiner algorithm to fix this problem (Méger and Rigotti, 2004).

An extension of Episodes expressed as temporal logic programs with operators like *until* or *since* is described in (Padmanabhan and Tuzhilin, 1996). A generative approach to episode mining is described in (Mannila and Meek, 2000). (Harms et al., 2001) uses closed sets of Episodes to create representative Episode rules. Efficient matching of found Episodes in new data can be done using directed acyclic subsequence graphs (Troníček, 2001).

In (Harms et al., 2002) the author presents a method mining for Episode association rules using minimal occurrences with constraints and time lag (MOWCATL) from multivariate symbolic time series or symbolic time sequences (Harms et al., 2002; Harms and Deogun, 2004). Instead of generating Episode rules in a post processing step from frequent Episode patterns, the algorithms directly mines rules with an antecedent part and a consequent part separated by a time lag. This way the rules can always be used for prediction. Separate maximum length constraints can be specified for the both rule parts. An example rule involving a serial Episode in the antecedent and a parallel Episode in the consequent could be: *if A at time t is followed by B before $t+\delta_1$ then C and D occur in a window of length δ_2 before $t+\delta_3$*. Further constraints can be integrated to restrict the search to interesting rules. In addition to order and concurrency, synchronicity is supported by so called combined event types in multivariate time series. For

each combination of the symbols from different dimensions of the time series occuring at the same time point, an additional event symbol is generated.

Casas-Garriga describes a different way of finding partial orders in a set of sequences, the so-called transactions. First, an algorithm for mining closed sequential patterns is applied (Yan et al., 2003; Wang and Han, 2004; Tzvetkov et al., 2003). The resulting sequential patterns are closed in the sense that there is no super-pattern with the same support. They do not, however, form a lattice structure as closed itemsets do, because there can be sequential patterns occuring in the same transactions that are not contained in one another. In the next step pairs consisting of a set of closed sequential patterns and a set of transactions in which all these patterns occur are formed. The algorithm ensures maximality of these pairs in the sense that no additional sequence occurs in all the transactions and there is no additional transaction in which all the sequences occur. These maximal pairs form a lattice structure and each of them is converted into a partial order resulting in a lattice of closed Episodes.

In (Mooney and Roddick, 2004) Episodes are combined with a subset of Allen's relations to express more temporal relations among Episodes than co-occurrence within a window. First, frequent episode patterns are mined. Then within each Episode so-called interacting Episodes are searched, that is sub patterns of the Episode for which the during, overlaps, or meets relation holds. This doesn't really add any new temporal concepts according to our definition (see Chapter 3.3), but increases the expressiveness of the resulting Episode rules. Given the frequent serial Episode *A followed by B followed by C followed by D* an example rule reads *(A followed by C) overlaps (B followed by D).*

The first approach for partial periodic patterns in symbolic time series is described in (Han et al., 1998). Partial periodicity means that the repetition does not necessarily occur on the whole time series, but only on segments. Given a period length, patterns are mined with an Apriori algorithm. For a period k the patterns consist of k positions filled with symbols or wild cards and cover the temporal concepts of order and periodicity. Yang et al. also mine the period along with the patterns (Yang et al., 2000). The method allows missing occurrences as well as shifted periods between segments of occurrence. In subsequent work surprising patterns are mined using an information measure rather than support (Yang et al., 2001b, 2002). Recently, extensions for more robust pattern matching allowing random replacements (Yang et al., 2003) and meta patterns (Yang et al., 2004) have been proposed. In (Berberidis et al., 2002) the Fast Fourier Transform (FFT, Shatkay (1995b)) of binary vectors for each symbol is used to obtain candidates for the algorithm of (Han et al., 1998). In contrast (Elfeky et al., 2004) uses a single pass of the data to mine periodic patterns with a similar method. An incremental version of (Han et al., 1998) is proposed in (Aref et al., 2004). A different approach to periodicity mining based on inter arrival times of a symbol is presented in (Ma and Hellerstein, 2001). Mining candidate periods before association rules is fast, while the opposite is more robust, because random occurrences of a pattern are less likely than those of a single symbol. Periodical patterns are typically searched in retail data, where discovered periodicity can be used for supply chain management or advertisement.

Saetrom and Hetland criticize the restrictive rule languages, e.g., for Episodes, needed to make many mining approaches feasible. The Interagon Query Language (IQL, Interagon (2002)), a general rule language that includes many others as special cases, is proposed. It is defined over univariate symbolic time series and symbolic time sequences. It covers the concepts of order, concurrency, and duration. Duration is modelled by allowing repetition of symbols. Similar to (Harms et al., 2002) a time lag between the antecedent and the consequent of a rule is allowed. The patterns are mined using Genetic Programming (Koza, 1998) where each individual is the syntax tree of a candidate pattern. The candidates are evaluated using special hardware to speed up the search. The patterns are mined directly optimizing an interestingness measure, instead of the more commonly used support and confidence combination. It is mainly the need

for special hardware that restricts this approach. It would be interesting to see, whether it is still feasible with standard, possibly parallel, hardware.

A hierarchical method that is less concerned with implication rules but more with the recognition of typical high level states comes from the field of ubiquitous computing (Schmitt, 2002). The goal is to recognize the context of the user of a mobile phone from sensor measurements integrated in the phone (Himberg et al., 2003). The symbolic vectors from a multivariate symbolic time series are clustered with the Symbol Clustering Map (SCM, (Flanagan, 2003)). This expresses the concept of synchronicity and creates a univariate symbolic time series. Repeating symbols are dropped, therefore neglecting concept of duration at this level. A triplet encoding of the symbols is again clustered with the SCM to obtain patterns similar to Episodes representing a mixture of the concepts order and concurrency. Consecutive symbols of the same cluster label somewhat represent the concept of duration. Many of the scenarios present in the training data were successfully recognized. More details on the method are given in (Mäntyjärvi, 2003).

The approaches for rule discovery in symbolic time series and sequences are summarized in Table 6.1 roughly ordered by increasing expressivity of the rule language. For several contributing authors the earlier publication is listed. The first four methods can only model the concept of order. The Semantic Patterns do not allow gaps. The wild cards of the approach in (Vilo, 1998) can model short gaps. The Association rules of (Das et al., 1998) explicitly model gaps, but use only two symbols per rule. The patterns of MSDD allow gaps via wild cards, but also relate only two points in time.

Author(s)/Year	Method/Keyword	Data model			Temporal concepts				
		univariate symbolic time series	multivariate symbolic time series	symbolic time sequence	duration	order	periodicity	concurrency	synchronicity
Vilo (1998)	Suffix Trie	×				×			
Cohen et al. (2001)	Semantic Patterns	×				×			
Das et al. (1998)	Associations	×	×	×		×			
Oates et al. (1996)	MSDD	×	×			×			
Oates et al. (1997)	MEDD	×	×	×				×	
Mannila et al. (1995)	Episodes	×	×	×		×		×	
Harms et al. (2002)	MOWCATL	×	×	×		×		×	×
Mooney et al. (2004)	Interacting Episodes	×	×	×		×		×	
Han et al. (1998)	Partial Periodicity	×				×	×		
Saetrom et al. (2003)	IQL	×		×	×	×		×	
Himberg et al. (2003)	SCM		×		×	×		×	×

Table 6.1: Categorization of unsupervised rule mining methods based on time point data models, i.e., symbolic time series and sequences.

The concept of concurrency is important for time sequences and discovered by the next four methods starting with MEDD. All except MEDD also represent order. In contrast to traditional Episode rules, MOWCATL is specifically designed to produce prediction rules and introduces a method to represent synchronicity of symbols from multivariate time series. All methods for

symbolic time sequences can also be applied to univariate symbolic time series. An extension to multivariate symbolic time series is usually possible as well.

Periodical patterns include the concept of order within a pattern. Periodicity is not commonly mined from time sequences. The IQL approach offers a rich rule language that even represents duration of symbols by modelling repetition. It is not clear, however, whether it is easily extendable to multivariate time series. The SCM method offers the most temporal concepts, but unfortunately they are not clearly separated in the method. Duration is partly neglected and partly represented in the results. Only local order represented by the triplet encoding. Clustering the triplets represents concurrency.

6.2 Time Interval Rules

The following approaches are all based on interval data models. This implicitly models the concept of duration. The length of the intervals, however, is not always used.

Villafane et al. search for containments of intervals in a multivariate symbolic interval series (Villafane et al., 1999). This represents the temporal concept of coincidence. A containment lattice is constructed from the intervals and rules are mined with the so called Growing Snake Traversal to reduce the storage space required by the naive algorithm. The following example rule is given *"when network usage is at or above 55%, disk usage is at or above 40%, and when such disk usage is observed, CPU usage during that time dips below 30%"* (Villafane et al., 1999). The duration of the intervals is not used.

Last et al. mine rules in a univariate contiguous numeric interval series (Last et al., 2001). Each interval is associated with several numeric or symbolic features and a single symbolic target attribute. Association rules on pairs of adjacent intervals are mined using the Info-Fuzzy Network (IFN, Maimon and Last (2000)), an integrated feature selection and association rule mining algorithm without candidate generation using conditional mutual information. Since the number of rules with quantitative features is usually large, the rule set is reduced using fuzzy theory. The final rules are association rules on labeled intervals interpreted as prediction rules for the symbolic target attribute of the second interval in a pair. An example from weather analysis reads: *"if this years precipitation is decreasing and was high, it is likely to be low next year"* (Last et al., 2001). Even though several features per interval are used, the interval series itself is univariate. The only temporal concept directly expressed by the rules is order. Duration can be explicitly expressed by using it as a numerical feature for the intervals.

In (Kam and Fu, 2000) interval rules are composed with Allen's relations and searched with an Apriori algorithm. The rule language therefore covers the concepts of coincidence, synchronicity, and order. The duration of an interval is not explicitly used. The search space is restricted to right concatenation of intervals to existing patterns, so-called A1 patterns, and by a threshold for the maximum length of a pattern. An example pattern is *"(((A overlaps B) before C) overlaps D)"*. No implication rules are generated. The underlying data model is a set of interval sequences. The support of a pattern is determined by counting the sequences of a set of sequences in which it appears. This can easily be generalized to counting the occurrences within sliding windows of a single long interval sequence.

In (Cohen, 2001) a subset of Allen's relations is used to mine association rules, called Fluents, in multivariate contiguous binary interval time series (Cohen, 2001; Cohen et al., 2002). This data model can also be interpreted as a symbolic interval sequence. The meets and before relations are merged into one. All three concepts expressible by Allen's relations are covered, but duration is not modelled. Significance of associations is determined with contingency tables similar to (Oates et al., 1997), thus based on counting occurrences as in (Kam and Fu, 2000). During learning the search is restricted to a sliding window and composites of single intervals and/or already significant patterns. The associations over time are interpreted as cause and

effect rules, an example from robot data reads: *"if the gripper is closed while an object is in the gripper then the holding object state will be observed next"* (Cohen, 2001).

The TCon method for temporal knowledge conversion presented in (Guimarães, 1998; Guimarães and Ultsch, 1999; Guimarães et al., 2001) does not use Allen's interval relations. They are criticized for the strictness w.r.t. interval boundaries. The patterns are expressed with the *Unification-based Temporal Grammar* (UTG) (Ultsch, 1996b; Guimarães and Ultsch, 1997; Ultsch, 2004), a hierarchical rule language developed for the description of patterns in multivariate time series. So-called Temporal Complex Patterns are constructed via several abstraction levels. At the level corresponding to the data model of symbolic interval series, they consider patterns of several intervals that are *more or less simultaneous*, called Events. The duration of Events is modelled explicitly recording minimum, maximum, and mean duration. The significance of Events is measured by conditional probability estimates. Since interval lengths are also considered, this covers the temporal concepts of duration and synchronicity. The next level expresses the concept of order among Events with so-called Sequences. A distinction between *follows* and *immediately follows* is made. At each level a linguistic representation with temporal grammatical rules is provided to enable the interpretation by experts. An example rule from the analysis of sleeping disorders reads *"no air flow and no respiratory movement without snoring is followed by no air flow and reduced chest and no abdominal movements without snoring is after at most five seconds followed by strong breathing with snoring"* (Guimarães, 1998). The results of this analysis are validated with a questionnaire of experts and rule based systems (Guimarães, 1998; Guimarães et al., 2001). Most steps of TCon are performed manually based on visualizations and statistical summaries of the patterns of the previous levels, no algorithms for the automatic discovery of the temporal concepts of synchronicity and order are described. U-Matrix (Ultsch, 1993) visualizations of ESOM Ultsch and Mörchen (2005) are used to aid the discovery of symbolic states from the numerical time series and to group similar Event patterns (Guimarães, 1998; Ultsch, 1999). The temporal concept of coincidence is not expressible by the UTG patterns, because the intervals are required to start and end almost simultaneously.

Another method using Allen's relations to mine rules in symbolic interval sequences is described in (Höppner, 2001, 2003). The search space is restricted by using a sliding window. The patterns are mined with an Apriori algorithm. In contrast to (Kam and Fu, 2000) implication rules are generated and support and confidence are not based on counting occurrences but rather on the lifetime of a pattern within the sliding window. The rule generation from frequent patterns works similar to association rules, but only forward in time to ensure the use for prediction. An example rule reads: *if (A starts at time t, ends before B starts and the gap is covered by C) then (B overlaps D) occurs before* $t + \delta$. The usage of quantitative attributes, e.g., the length of the intervals or gaps and ranking of the rules by the J-measure are proposed in (Höppner and Klawonn, 2002). This way duration is expressible in addition to the concepts of coincidence, synchronicity, and order covered by Allen's relations. Further extensions for handling feature ambiguity (Höppner, 2002b) and rule set condensation (Höppner, 2002a) are proposed.

In Table 6.2 the properties of the described approaches for rule discovery in interval series and sequences are listed in order of increasing expressivity of the rule language and earlier publication. All except the second method work on multivariate interval series and the almost equivalent data type of interval sequences. The first two methods are rather simple in that they only model few temporal concepts. (Last et al., 2001) is further restricted by the need to specify a target attribute and the use of a contiguous interval series. Allen's interval relations provide a higher temporal expressivity of the resulting patterns and rules. In contrast to (Höppner, 2001) the methods of (Kam and Fu, 2000) and (Cohen, 2001) do not model duration. The UTG/TCon also offers duration but not coincidence.

Author(s)/Year	Method/Keyword	Data model			Temporal concepts			
		univariate numeric interval series	multivariate symbolic interval series	symbolic interval sequence	duration	order	coincidence	synchronicity
Villafane et al. (1999)	Containments		×	×			×	
Last et al. (2001)	IFN	×			×	×		
Kam and Fu (2000)	Allen/A1		×	×		×	×	×
Cohen (2001)	Allen/Fluents		×	×		×	×	×
Guimarães and Ultsch (1999)	UTG/TCon		×	×	×	×		×
Höppner (2001)	Allen/Window		×	×	×	×	×	×

Table 6.2: Categorization of unsupervised rule mining methods based on time interval data models, i.e., symbolic interval series and sequences.

Chapter 7

Time Series Knowledge Representation

We present the Time Series Knowledge Representation (TSKR), a pattern language for knowledge discovery from multivariate time series and symbolic interval data. We first motivate the need for a new flexible but concise way of describing temporal phenomena in Chapter 7.1 by pointing out deficiencies of existing methods through examples. We then proceed with the definition of the data models and the pattern language for the representation of duration, coincidence, and partial order in Chapter 7.2. The TSKR is analyzed and compared to Allen's relations and the UTG in Chapter 7.3. Efficient algorithms to mine the TSKR patterns are presented in Chapter 8 completing the Time Series Knowledge Mining methodology.

7.1 Motivation

The aim of this thesis is to develop a methodology for the unsupervised discovery of interpretable patterns from numeric multivariate time series supporting knowledge discovery tasks. The representation of temporal knowledge should be rich in temporal expressivity covering many temporal concepts. The patterns should be easily comprehensible by humans to support manual analysis. For unsupervised mining only the subsequent human interpretation of the results can determine the novelty and usefulness. Symbolic data models should be used, as each symbol is commonly associated with a qualitative description of parts of the time series. Neither the size of the patterns, nor the amount of patterns should overwhelm the user. The presence of noise in the data should be graciously handled. Finally, it should be possible to efficiently mine the patterns with computer programs.

The most common symbolic temporal data models and expressible temporal concepts of existing rule generation approaches have been compared in Chapter 6. Time point based methods are not adequate for time series data with persisting states, because the concept of duration is not well representable. Only IQL can model duration by repetition. The most expressive time point based method SCM expresses a total of four temporal concepts, but it is not aimed at creating understandable rules.

For time interval data there are in principle two qualitatively different approaches that offer many temporal concepts: the rules based on Allen's relations and the UTG. We claim that both have major disadvantages in expressivity, robustness, or interpretability for mining patterns from time series. We briefly formulate and justify our claims, a more formal analysis is given in Chapter 7.3.

Claim 1. UTG patterns cannot express all patterns expressible with Allen's interval relations.

A simple example of a pattern with two overlapping intervals that has a long prefix and suffix where only one interval is present is shown in Figure 7.1. This cannot be expressed by a combination of UTG Events and Sequences (Chapter 6.2) under all sensible choices of the threshold parameter for Events determining almost synchronicity. The distance between the two start points (or end points) would need to be much shorter than the duration of the overlapping part for this pattern to be considered almost synchronous. An extension to cover the concept of coincidence is needed.

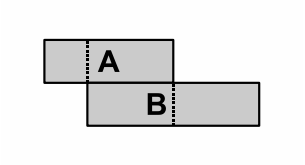
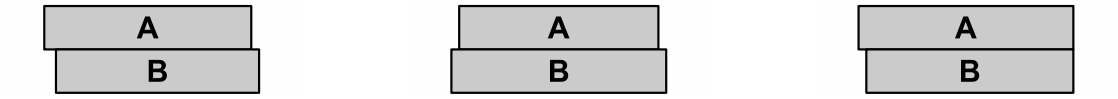


Figure 7.1: Example of an interval pattern expressible with Allen's *overlaps* relations but not with the UTG.

Claim 2. Patterns expressed with Allen's interval relations are not robust.

Several of Allen's relations require the equality of two or more interval endpoints. If the original time series data is collected from sensors, noise is almost always present and hard to be removed completely. Any interval sequence obtained by discretization or segmentation will be inherently noisy w.r.t. the start and end points of the intervals. Noise in the relative positions of interval boundaries can lead to patterns that intuitively describe the same relationship between intervals, but are fragmented into different relations as defined by Allen. Figure 7.2 shows three examples of almost equal intervals represented by different relations.



(a) overlaps (b) during (c) finishes

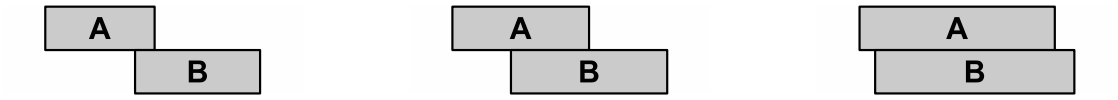
Figure 7.2: Three different relations of Allen represent fragments of a pattern with almost equals intervals.

Claim 3. Patterns expressed with Allen's interval relations are ambiguous.

The same relation of Allen's interval relations can represent intuitively very different situations. In Figure 7.3 three intuitively very different versions of the *overlaps* relation are shown as an example. Even more ambiguous is the compact representation of patterns based on Allen's relations (Kam and Fu, 2000; Cohen, 2001), because different descriptions are valid for the exact same pattern (see Chapter 7.3).

Claim 4. Patterns expressed with Allen's interval relations are not easily comprehensible.

The patterns in (Höppner, 2001) are expressed by listing the pairwise relations of all intervals. This can quickly lead to lengthy pattern descriptions with $\frac{k(k-1)}{2}$ pairwise relations for a pattern with k intervals.



(a) small overlap (b) medium overlap (c) large overlap

Figure 7.3: Three different instances of Allen's *overlaps* relation that have quite distinct appearances.

In summary, Allen's relations seem not well suited for mining patterns from imprecise interval data, because they are ambiguous, not robust, and potentially hard to interpret. The UTG Event patterns are designed to be more robust, by matching interval endpoint with a threshold. In contrast, the distinction between two versions of the *followed by* operator in UTG Sequences is also sensitive to small disturbances of interval endpoints. Event pattern cannot express coincidence and the restriction to a fixed number of intervals further limits the search space. The hierarchical structure of the UTG and the separation of temporal concepts over several mining steps, however, offers advantages for the knowledge discovery process. The intermediate patterns can be analyzed before more complex patterns are searched. The TSKR utilizes these core ideas (Ultsch, 1996b, 2004) inheriting the advantages while fixing the identified discrepancies of the UTG. We define a hierarchical pattern language that covers the temporal concepts of coincidence and partial order achieving higher expressivity than the UTG and Allen's relations.

7.2 Definitions

The TSKR consists of a hierarchy of patterns each describing single temporal concepts. Patterns are built in a bottom-up process. Each level of the hierarchy inhibits an increase in complexity and temporal abstraction compared to the previous level. In this Chapter the semantic of each pattern is defined. In order to make the patterns more easily comprehensible for interpretation, we further propose a rule syntax for the qualitative description as well as a graphical representation of the constructs through examples.

As a running example we will use the Skating data set, described in more detail in Chapter A.1. It consists of physiological and mechanical measurements of professional Inline Speed Skating under controlled conditions.

7.2.1 Data Models

We first define the data models used in the different mining stages. The input time series are required to be uniformly sampled. We assume an underlying process generating the time series that is observed at discrete time points.

Definition 7.1. A *Chronon* $\tau \in \mathbb{R}$ is the sample interval.

Definition 7.2. A uniformly spaced *series of time points* of length N w.r.t. a Chronon τ and a start time $\tau_0 \in \mathbb{R}$ is the finite set $\{\theta_i = \tau_0 + i\tau, i \in \mathbb{X}\}$ where $\mathbb{X} = \{1, ..., N\} \subset \mathbb{N}$ are the indices i of the time points are called *ticks*.

Without loss of generality, we will define the following data models and patterns based on the natural numbering of the time points, the ticks $\mathbb{X}$.

Definition 7.3. A *time interval* is a pair $[s, e] \in \mathbb{X}^2$ with $s \leq e$ that corresponds to the interval $[\theta_s, \theta_{e+1}) \subset \mathbb{R}$. The finite set of all time intervals over $\mathbb{X}$ is noted $\mathbb{I} = \{[s, e] \in \mathbb{X}^2 | s \leq e\}$. We

write $[s, e] \subseteq [s', e']$ if $s' \leq s$ and $e \leq e'$ with equality if and only if $s = s'$ and $e = e'$. The duration of an interval in units of ticks is $d([s, e]) = e - s + 1$ and in units of time $\tau d([s, e])$.

Definition 7.4. Two interval $[s, e]$ and $[s', e']$ *overlap* if $\{s, ..., e\} \cap \{s', ..., e'\} \neq \emptyset$.

Definition 7.5. We define a *partial order* $\prec$ of intervals as $[s_1, e_1] \prec [s_2, e_2] \Leftrightarrow e_1 < s_2$. We say that $[s_1, e_1]$ is *before* $[s_2, e_2]$.

Note that our notion of partial order among intervals considers overlapping intervals as not related. Figure 7.4(a) shows an example of two related intervals, while in Figure 7.4(b) A and B are not related.

(a) $A \prec B$ (b) $A \not\prec B \wedge A \not\succ B$

Figure 7.4: Two examples demonstrating the partial order on intervals. Intervals that overlap are not related.

Let Σ be a finite set of *symbols* σ. We attach symbols to intervals, representing the pattern classes described below.

Definition 7.6. A *symbolic time interval* is a triple $[\sigma, s, e]$ with $\sigma \in \Sigma$ and $[s, e] \in \mathbb{I}$.

Definition 7.7. A *symbolic interval series* is a finite set of non overlapping symbolic time intervals $\mathcal{I} = \{[\sigma_i, s_i, e_i] | \sigma_i \in \Sigma, [s_i, e_i] \in \mathbb{I}, i = 1, ..., N; e_j < s_{j+1}, j = 1, ..., N-1\}$. The duration of a symbolic interval series is $d(\mathcal{I}) = \Sigma_{i=1}^{N} d([s_i, e_i])$.

Definition 7.8. A *symbolic interval sequence* is a finite set of symbolic time intervals $\mathcal{I} = \{[\sigma_i, s_i, e_i] | \sigma_i \in \Sigma, [s_i, e_i] \in \mathbb{I}, i = 1, ..., N\}$

An interval sequence can contain overlapping intervals and equal time intervals. Exact duplicates of the symbolic intervals are excluded, however, intervals with the same start and end points are only allowed if their symbols differ.

Definition 7.9. A d-dimensional *time series* w.r.t. a series of time points $\mathbb{X} = \{x_1, ..., x_N\}$ of length $N \in \mathbb{N}$ is $Y = \{(x_i, y_i) | y_i = \langle y_{i,1}, ..., y_{i,d} \rangle^T \in \mathbb{R}^d, x_i \in \mathbb{X}, i = 1, ..., N\}$. If $d = 1$, Y is called *univariate*, for $d > 1$ it is a *multivariate* time series.

Definition 7.10. An *itemset* is a subset $S = \{\sigma_1, ..., \sigma_n\} \subseteq \Sigma$ of all symbols.

Definition 7.11. An *itemset sequence* is a vector of itemsets $s = \langle S_1, ..., S_k \rangle$ with $S_i \subseteq \Sigma$ for $i = 1, ..., k$. We say that s is a *subsequence* of $s' = \langle S'_1, ..., S'_l \rangle$ and write $s \subseteq s'$ if $k \leq l$ and $\exists j_1 < ... < j_k$ such that $S_i \subseteq S'_{j_i}$ for $i = 1, ..., k$.

Definition 7.12. We write $s \diamond s' = \langle S_1, ..., S_k, S'_1, ..., S'_l \rangle$ for the *concatenation* of the itemset sequences $s = \langle S_1, ..., S_k \rangle$ and $s' = \langle S'_1, ..., S'_l \rangle$.

Definition 7.13. An *itemset time interval* is a triple $[S, s, e]$ with $S \subseteq \Sigma$ and $[s, e] \in \mathbb{I}$.

Definition 7.14. An *itemset interval series* is a finite set of non overlapping itemset time intervals $\mathcal{I} = \{[S_i, s_i, e_i] | S_i \subseteq \Sigma, [s_i, e_i] \in \mathbb{I}, i = 1, ..., N; e_j < s_{j+1}, j = 1, ..., N-1\}$. The duration of an itemset interval series is $d(\mathcal{I}) = \Sigma_{i=1}^{N} d([s_i, e_i])$. We write $\mathcal{I}|_{[s,e]} = \{[S_j, s_j, e_j] | [S_j, s_j, e_j] \in \mathcal{I} \wedge s_j \geq s \wedge e_j \leq e\}$ for all itemset intervals valid during the interval $[s, e]$. If used in the context of itemset sequences, an itemset interval series is reduced to the itemsets sequence $\langle S_1, ..., S_N \rangle$, dropping the information on the time intervals.

Definition 7.15. The support of an itemset sequence s in an itemset interval series $\mathcal{I}$ w.r.t. a time interval sequence W is $sup_W(s) = |\{w|w \in W \wedge s \subseteq \mathcal{I}|_w\}|$, i.e., the number of intervals from W in which it occurs.

7.2.2 Pattern Model

We now turn to the core structure of the knowledge representation, namely the patterns and the occurrences of patterns in the input data.

Definition 7.16. Let Λ be a finite set of *labels* λ. Let $l : \Sigma \mapsto \Lambda$ be the function assigning each symbol a label. Let Φ be a set of characteristic functions $\phi_{\mathcal{X}} : \mathbb{I} \rightarrow \{\textsc{True}, \textsc{False}\}$, where $\mathcal{X}$ is some arbitrary input data. A *pattern* $(\sigma, \lambda, \phi_{\mathcal{X}})$ is a semiotic triple (Ultsch, 1996b) composed of:

- $\sigma \in \Sigma$, a unique symbol representing the pattern in higher levels constructs (syntax),
- $\lambda = l(\sigma) \in \Lambda$, a label providing a textual description of the practical meaning of the pattern (pragmatic),
- $\phi_{\mathcal{X}} \in \Phi$, a characteristic function determining when the pattern occurs (semantic).

The semantic of ϕ and the data type of $\mathcal{X}$ will differ for the various mining tasks defined below. We write ϕ for $\phi = \textsc{True}$ and $\neg\phi$ for $\phi = \textsc{False}$.

The concept of semiotic triples is consistently used on all levels of the TSKR pattern hierarchy. It enables the pragmatic annotation of each pattern with labels to aid the later interpretation when used as parts of larger patterns.

Definition 7.17. An *occurrence* of a pattern is an interval $[s, e] \in \mathbb{I}$ with $\phi_{\mathcal{X}}([s, e])$. A *maximal occurrence* is an occurrence $[s, e]$ such that $\forall [s', e'] \supset [s, e]\ \neg\phi_{\mathcal{X}}([s', e'])$. A *marginally interrupted occurrence* is an interval $[s, e]$ such that

$$\exists s < e' < s' < e \text{ with } \phi_{\mathcal{X}}([s, e']) \wedge \phi_{\mathcal{X}}([s', e]) \tag{7.1}$$

and

$$\frac{d(\{[s'', e'']|[s'', e''] \subset [s, e], \phi_{\mathcal{X}}([s'', e'']), [s'', e''] \text{ maximal}\})}{d([s, e])} \geq 1 - \alpha \tag{7.2}$$

and

$$max(\{d([s'', e'])|[s'', e''] \subset [s, e], \neg\phi_{\mathcal{X}}([s'', e'])\}) \leq d_{max} \tag{7.3}$$

where $\alpha < 1$ is a parameter specifying the maximum percentage of the total length of all allowed interruptions and d_{max} is a parameter specifying the maximum absolute length of a single interruption.

A marginally interrupted occurrence is thus required to start and end with a maximal occurrence (Property 7.1).

7.2.3 Grouping of input data (Aspects)

The input data model for the first mining step is a multivariate time series of possibly high dimensionality. Following the divide-and-conquer approach the dimensions of the time series should be split into several groups for separate processing during the first few mining steps.

Definition 7.18. An *Aspect* $A = (\lambda, V)$ of a time series Y is a named a-dimensional time series of the same length but with possibly reduced dimensionality, i.e., $V = \{(x_i, v_i) | v_i = \langle v_1, ..., v_a \rangle^T = \langle y_{i,j_1}, ..., y_{i,j_a} \rangle^T \in \mathbb{R}^a, j_1 < j_2 < ... < j_a, j_k \in \{1, ..., d\}, k = 1, ..., a, i = 1, ..., N\}$. We write $A \subset Y$ for an Aspect A of Y. If $a = 1$, A is called *univariate*, for $a > 1$ it is a *multivariate* Aspect. We write $A|_{[s,e]}$ for the Aspect with the sub-series of V on the interval $[s, e]$.

An Aspect describes a semantic property of the multivariate input time series represented by a subset of the dimensions. This is particularly useful if the dimensionality is large and correlation exists within subsets of the variables. Having less semantical dimensions to consider in the further processing steps eases the interpretation of the patterns.

Example 7.1. For the Skating data set 11 time series were recorded with sensors, the labels of which are shown in the first column of Table 7.1. The first row corresponds to a pressure sensor in the shoe and was placed in a univariate Aspect. The next three rows are angle measurements at the joints of the leg. They describe the leg's movement and are grouped in a three dimensional Aspect. The last seven rows describe the activity of different muscles of the skater's leg. The grouping of the muscles can be done using the physiological functions. The third and forth column show a coarse grouping into flexor and extensor muscles. The last six columns show a more fine grained grouping of flexor and extensor functionality for the three joints. Note, that several muscles serve more than one purpose and make some Aspects overlap, e.g., the Tensor Fasciae Latae muscle in the last row is an extensor for the knee and a flexor for the hip. A primary and secondary function may be described, the latter is parenthesized in Table 7.1.

Time Series Variables	Aspects									
	Foot contact	Leg position	Flexor muscles	Extensor muscles	Ankle flexors	Ankle extensors	Knee flexors	Knee extensors	Hip flexors	Hip extensors
Foot contact	×									
Ankle angle		×								
Knee angle		×								
Hip angle		×								
Tibialis Anterior muscle			×		×					
Medial Gastrocnemius muscle			(×)	×		×	×			
Vastus Medialis muscle				×				×		
Rectus Femoris muscle			(×)	×				×	×	
Semitendinosus muscle			×	(×)			×			×
Gluteus Maximus muscle				×				×		×
Tensor Fasciae Latae muscle			(×)	×				×	×	

Table 7.1: Possible groupings of the variables from the Skating data into Aspects. The muscle activities can be grouped on different abstraction levels, e.g., using only their main functionality (flexor or extensor), additionally considering the involved joints (ankle, knee, and hip), or simply using seven univariate Aspects.

7.2.4 Representing duration (Tones)

The first pattern type covers the temporal concept of duration. It represents a persisting state or property present in the input time series with a symbolic interval.

Definition 7.19. A *Tone*[1] pattern is a semiotic triple $t = (\sigma, \lambda, \phi_A)$ with $\sigma \in \Sigma$, $\lambda \in \Lambda$, and a characteristic function $\phi_A \in \Phi$ indicating the occurrence of a state in an Aspect A on a given time interval. Let the set of all Tones be $\mathbb{T}$.

We explicitly allow arbitrary characteristic functions for detecting a property of the input time series, but the preference is on easily interpretable functions. For univariate time series we give some examples that list common notions of persisting properties in the numerical time series.

Example 7.2. A *value-based* Tone for a univariate Aspect can be specified with an interval condition on the values of the time series:

$$\phi_A([s,e]) \leftarrow v_{min} < v_i \leq v_{max}\ i = s, ..., e \tag{7.4}$$

Example 7.3. A textual rule for a value-based Tone from the Skating data set is given in Figure 7.5. The muscle activity of the Gluteus Maximus muscle is defined to be high if the time series values rise above 0.7. The label is chosen as a compact summary of the rule. For the graphical representation of a Tone we simple use a box with the label as shown in Figure 7.6.

'GM is high' *when* 'Activity of Gluteus Maximus muscle' *is greater than* 0.7.

Figure 7.5: Textual representation of a Tone from the Skating data. The activity of the muscle is defined as high if it is larger than a threshold.

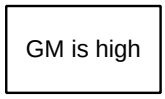


Figure 7.6: Graphical representation of a Tone from the Skating data.

Example 7.4. A *trend-based* Tone for a univariate Aspect can be specified with an interval condition on the slope of an interpolation of the values. Let $\mathcal{S}$ be an interval series representing a segmentation of the Aspect.

$$\phi_A([s,e]) \leftarrow \exists [s',e'] \in \mathcal{S}\ [s,e] \supseteq [s',e'] \text{ s.t. } b_{min} < \hat{b} \leq b_{max} \text{ where } \hat{b} = \frac{v'_e - v'_s}{e' - s'} \tag{7.5}$$

Note that a larger interval from the given segmentation is used for the definition of the trend-based Tone to ensure that the function holds on all subintervals of $[s,e]$. The same is needed for the following example.

Example 7.5. A *shape-based* Tone for a univariate Aspect A can be specified with a threshold on the distance between a sub-series and a prototype. Let $\mathcal{S}$ be an interval series representing a segmentation of the Aspect.

$$\phi_A([s,e]) \leftarrow \exists [s',e'] \in \mathcal{S}\ [s,e] \supseteq [s',e'] \text{ s.t. } D(A|_{[s',e']}, m) < D_{max} \tag{7.6}$$

where m is a prototypical time series and D is a shaped-based time series distance function (see Chapter 5.3).

[1] From Greek tonos; literally, act of stretching; vocal or musical sound of a specific quality; a sound of definite pitch and vibration (Merriam-Webster, 2005).

Value-based Tones for multivariate Aspects can be specified with a conjunction of interval conditions.

Example 7.6. A *value-based* Tone for a multivariate Aspect describing a hypercube in $\mathbb{R}^a$:

$$\phi_A([s,e]) \leftarrow v_{i,j} < v^j_{max} \wedge v_{i,j} > v^j_{min}\ i = s, ..., e; j = 1, ..., a \tag{7.7}$$

7.2.5 Representing coincidence (Chords)

The second pattern type expresses the temporal concept of coincidence. An interval where several Tones occur simultaneously, is described by a new pattern.

Definition 7.20. A *Chord*[2] pattern is a semiotic triple $c = (\sigma, \lambda, \phi^T_{\mathcal{T}})$ with $\sigma \in \Sigma$, $\lambda \in \Lambda$, and a characteristic function $\phi^T_{\mathcal{T}} \in \Phi$ indicating the simultaneous occurrences of k Tones $T = \{t_i = (\sigma_i, \lambda_i, \phi_i) | i = 1, ..., k, k > 0\}$ on a given time interval according to the interval series $\mathcal{T}$ with occurrences of the Tones T:

$$\phi^T_{\mathcal{T}}([s,e]) \leftarrow \phi_i([s,e])\ i = 1, ..., k \tag{7.8}$$

A Chord consisting of k Tones is called a k-Chord, k is the *size* of the Chord. A 1-Chord is simply a copy of a Tone and is called a *trivial* Chord. Let the set of all Chords be $\mathbb{C}$.

Example 7.7. A textual rule for a Chord from the Skating data set is listed Figure 7.7. The *Active gliding* phase during Inline skating is characterized by the foot touching the ground, the leg's position corresponding to the gliding movement phase, and two major leg muscles showing high activity. For the graphical representation of a Chord we stack the Tone boxes with the label of the Chord on top as shown in Figure 7.8.

Chord 'Active gliding' *when* 'Foot on ground' and 'Gliding movement phase' and 'Vastus is high' and 'Gluteus is high' *coincide*.

Figure 7.7: Textual representation of a Chord from the Skating data. The foot contact during the gliding phase coincides with high activity of two muscles.

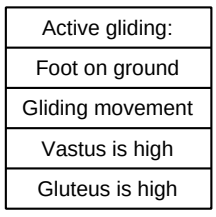


Figure 7.8: Graphical representation of a Chord from the Skating data.

Definition 7.21. We define a *partial order* $\subset$ on $\mathbb{C}$ such that $c_i \subset c_j \Leftrightarrow T_i \subset T_j$ where $c_i = (\sigma_i, \lambda_i, \phi^{T_i}_{\mathcal{T}})$ and $c_j = (\sigma_j, \lambda_j, \phi^{T_j}_{\mathcal{T}})$, i.e., c_i describes the coincidence of a subset of the Tones from c_j. We say that c_i is a *sub*-Chord of c_j and c_j is a *super*-Chord of c_i. We write the combination of two Chords obtained by merging their respective sets of Tones $c_i \cup c_j$.

[2]Three or more musical tones sounded simultaneously (Merriam-Webster, 2005).

Example 7.8. The Chord listed in Figure 7.9 is a sub-Chord of the Chord from Figure 7.7. Only one muscle is considered. This means, that each instance of this smaller Chord can possibly have a longer duration and that there can be more instances, because it is less restrictive than the Chord in Figure 7.7 involving four aspects. In the graphical representation of Chords we indicate this relationship by a line connecting the two Chords in a top down manner as shown in Figure 7.10.

Chord 'Gliding' *when* 'Foot on ground' and 'Gliding movement phase' and 'Vastus is high' *coincide*.

Figure 7.9: Textual representation of a sub-Chord of Figure 7.7 from the Skating data.

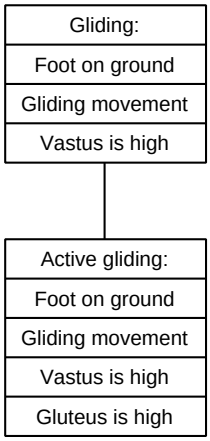


Figure 7.10: Graphical representation of a sub-Chord relationship for two Chords from the Skating data.

Definition 7.22. Let the *support set* of a Chord $c = (\sigma, \lambda, \phi)$ be the symbolic interval series of all maximal occurrences with duration of at least δ, written $S_\delta(c) = \{[\sigma, s, e] | [s, e] \in \mathbb{I}, \phi_{\mathcal{I}}^{T}([s, e]), [s, e] \text{ maximal}, d([s, e]) \geq \delta\}$. The *support* of a Chord is $sup_\delta(P) = d(S_\delta(c))$, i.e., the duration of the support set.

The occurrence of a Chord pattern implies the occurrence of all sub-Chords on the same interval. The number of sub-Chords grows quickly with the size of a Chord. Usually, larger Chords are more interesting, because they are more specific. If a sub-Chord only occurs whenever a certain super-Chord is also valid, it is redundant. It need not be listed separately in the result of mining Chords, because its presence is implied by the super-Chord. We therefore introduce the concept of closedness to provide a way to compactly and non redundantly represent a finite set of Chords. This is motivated by the concept of closedness for frequent itemsets (e.g. Pasquier et al. (1999); Boulicaut and Bykowski (2000); Pei et al. (2000); Zaki and Hsiao (2002)). The set of all closed frequent itemsets is lossless and can be orders of magnitudes smaller than all frequent itemsets. Just like itemsets represent a subset of a given set of items, Chords represent a subset of all symbols present in the symbolic interval series. We can thus easily transfer the concept of closedness to Chords.

Definition 7.23. A Chord c_i is *closed* if there are no super-Chords that have the same support, i.e., $\forall c_j \supset c_i, sup(c_j) < sup(c_i)$.

The definition of closedness considers a Chord as closed even if a larger Chord has only a slightly smaller support set. Two Chords could appear on almost the same intervals, with the intervals of the sub-Chord being only slightly longer in some places. Still, the smaller Chord would be considered closed in the sense of the definition above. With Tone patterns mined from possibly inexact and erroneous time series this is a harsh restriction. We therefore introduce the relaxed concept of margin-closedness to handle noisy interval data in a more robust way.

Definition 7.24. A Chord c_i is *margin-closed* w.r.t. a threshold $\alpha < 1$ if there are no super-Chords that have almost the same support, i.e., $\forall c_j \supset c_i, \frac{sup(c_j)}{sup(c_i)} < 1 - \alpha$.

The margin-closed Chords are a subset of the exactly closed Chords, because Definition 7.23 is stricter than Definition 7.24. There is a margin of support around each pattern, caused by pruning sub-patterns with similar support.

7.2.6 Representing partial order (Phrases)

The third pattern type expresses the temporal concept of partial order. Several Chords that occur in a particular partial order, are described by a new pattern.

Definition 7.25. A *Phrase*[3] pattern is a semiotic triple $p = (\sigma, \lambda, \phi_{\mathcal{C}}^{C,E})$ with $\sigma \in \Sigma$, $\lambda \in \Lambda$, and a characteristic function $\phi_{\mathcal{C}}^{C,E} \in \Phi$ indicating the occurrences of the k Chords $C = \{c_i = (\sigma_i, \lambda_i, \phi_i) | i = 1, ..., k, k > 0\}$ according to a partial order $E \subseteq \{\sigma_i\}^2$ on a given time interval.

$$\begin{aligned}
\phi_{\mathcal{C}}([s,e]) \quad \leftarrow \quad & (\forall i = 1, ..., k \; \exists [s_i, e_i] \subseteq [s,e] \; \phi_i([s,e])) && (7.9)\\
& \wedge (\exists i \in \{1, ..., k\} \; s_i = s) && (7.10)\\
& \wedge (\exists i \in \{1, ..., k\} \; e_i = e) && (7.11)\\
& \wedge (\forall i \neq j \in \{1, ..., k\}^2 \; [\sigma_i, s_i, e_i] \prec [\sigma_j, s_j, e_j] \Leftrightarrow (\sigma_i, \sigma_j) \in E) && (7.12)
\end{aligned}$$

A Phrase consisting of k Chords is called a k-Phrase, k is the size of the Phrase. Let the set of all Phrases be $\mathbb{P}$.

The characteristic function for a Phrase consists of four necessary conditions. Property 7.9 ensures that the intervals of all Chords are within the Phrase interval, while Property 7.10 and Property 7.11 prevent extra room before the first and after the last Chord, respectively. The occurrence of a Phrase is thus maximal by definition. A Phrase occurs on a particular interval but not on the sub-intervals. Property 7.12 requires the Chords to be in the partial order specified by E. Note, that any two intervals that have an order relation in E are not allowed to overlap. This restriction makes sense, because Chords already describe the concept of coincidence. Allowing overlapping Chords within a Phrase in general would mean to repeatedly represent the same concept and can in fact be equally represented by a larger Chord on the interval where two Chords overlap.

Example 7.9. Three overlapping Tones are shown at the top of Figure 7.11. The maximal Chords involving at least two Tones are shown below. The 2-Chords AB and BC overlap exactly on the occurrence interval of the 3-Chord ABC. Overlapping Chords are not allowed in Phrases if they are related by an order relation. Therefore only the first part of AB and the last part of BC are considered in the Phrase shown in the bottom row. This Phrase contains all information about the three Tones without containing overlapping Chords.

[3]Latin phrasis; from Greek phrazein: to point out, explain, tell; a characteristic manner or style of expression; a brief expression; a short musical thought typically two to four measures long closing with a cadence (Merriam-Webster, 2005).

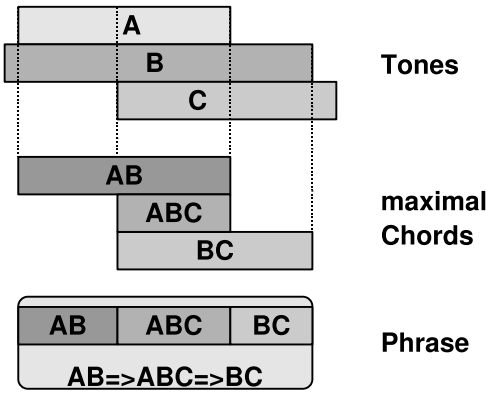


Figure 7.11: Maximal Chords obtained from Tones can overlap. Within a Phrase, only non-overlapping parts of Chords are used in the partial order relations.

Example 7.10. A Phrase from the Skating data set is shown in Figure 7.12. The order of the three Chords *Swing*, *Pre-activation*, and *Active gliding* was identified as the *Preparation* phase for the push-off during the repetitive skating movement. In the graphical representation we usually collapse the Chords to single blocks with only the label and indicate the order relationships with arrows in a left to right manner (see Figure 7.13).

Phrase 'Preparation' *when* 'Swing' *then* 'Pre-activation' *then* 'Active gliding'.

Figure 7.12: Textual representation of a Phrase for the Skating data.

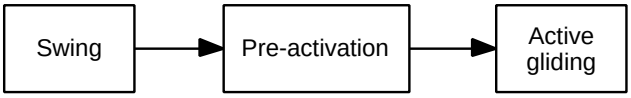


Figure 7.13: Graphical representation of a Phrase from the Skating data.

Definition 7.26. We define a *partial order* $\subset$ on $\mathbb{P}$ such that $p_i \subset p_j$ if and only if 1) $C_i \subset C_j$ where $p_i = (\sigma_i, \lambda_i, \phi_{\mathcal{C}}^{C_i,E_i})$ and $p_j = (\sigma_j, \lambda_j, \phi_{\mathcal{C}}^{C_j,E_j})$, i.e., p_i describes the partial order of a subset of the Chords of p_j and 2) $(\sigma_a, \sigma_b) \in E_i \Leftrightarrow (\sigma_a, \sigma_b) \in E_j$ where σ_a and σ_b are symbols of Chords from C_i, i.e., the common Chords must have the same partial order in both Phrases. We say that p_i is a sub-Phrase of p_j and p_j is a super-Phrase of p_i.

Example 7.11. The Phrase shown in Figure 7.15 is a super-Phrase of the Phrase from Figure 7.13. It offers more details by adding the Chord *Preparation of foot contact* and more order relations. No total order between the four distinct chords can be observed, but the total orders of three Chords each can be merged into a Phrase. For partial orders the graphical representation significantly aids the interpretation of the textual listing in Figure 7.14. Again, the relationship among sub- and super-Phrases is indicated by a vertical line.

Definition 7.27. Let the *support set* of a Phrase $p = (\sigma, \lambda, \phi_{\mathcal{C}}^{C,E})$ be the symbolic interval series of all (maximal) occurrences $S(p) = \{[\sigma, s, e] | [s, e] \in \mathbb{I}, \phi_{\mathcal{C}}^{C,E}([s, e])\}$. The *support* of a Phrase is $sup(p) = |S(p)|$, i.e., the number of occurrences.

The support of Phrases is determined by counting the occurrences, instead of summing up the durations of the maximal occurrence intervals as for Chords. This is necessary for the definition

Phrase 'Detailed preparation' *when*
'Preparation of foot contact' *then* 'Pre-activation' *then* 'Active gliding'
and 'Swing' *then* 'Pre-activation' *then* 'Active gliding'
and 'Swing' *then* 'Preparation of foot contact' *then* 'Active gliding'.

Figure 7.14: Textual representation of a super-Phrase of Figure 7.12 for the Skating data.

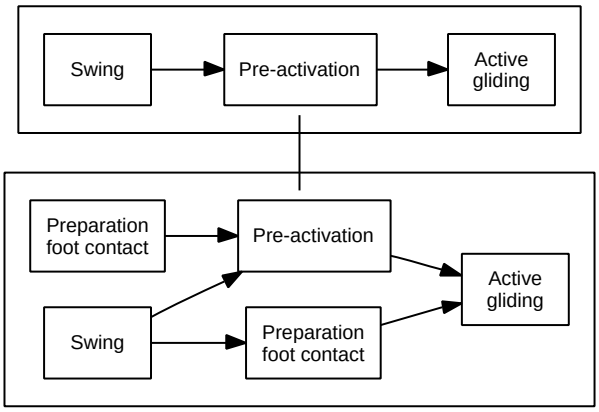


Figure 7.15: Graphical representation of a sub-Phrase relationship for two Phrases from the Skating data.

of closed Phrases below. Using the same notion as for Chords would not give an intuitive concept of closedness. Imagine an extension of a Phrase by an additional Chord occuring before the first Chord that is possible for each occurrence of the smaller Phrase pattern. The smaller pattern is redundant, because each time it appears, the larger pattern can also be observed. The occurrence intervals will all grow to cover the additional Chord up front, but the number of intervals will stay the same. The closedness of Phrases is thus defined using frequency, not duration. Just as Chords relate to itemsets, Phrases relate to sequential patterns in particular and partial orders in general and where closedness has been proposed recently (Yan et al., 2003; Casas-Garriga, 2005).

Definition 7.28. A Phrase p_i is *closed* if there are no super-Phrases that have the same support, i.e., $\forall p_j \supset p_i, sup(p_i) = sup(p_j)$.

The restriction for closedness is again rather strict. With large data set and many Phrase patterns found, a small difference in frequency does not justify keeping the smaller of two related patterns.

Definition 7.29. A Phrase p_i is *margin-closed* w.r.t. a threshold $\alpha < 1$ if there are no super-Phrases that have almost the same support, i.e., $\forall p_j \supset p_i, \frac{sup(p_j)}{sup(p_i)} < 1 - \alpha$.

The margin-closed Phrases are a subset of the exactly closed Phrases, because Definition 7.28 is stricter than Definition 7.29.

7.2.7 Summary of pattern hierarchy

The presented TSKR pattern language builds up an abstraction hierarchy in which the three temporal concepts duration, coincidence, and partial order are added at successive levels. The dimensions of a multivariate time series are first grouped into several semantically meaningful groups, the Aspects. The pattern hierarchy starts with Tones representing duration. Tones

describe time intervals where one Aspect persists in a qualitative state. Simultaneously occuring Tones form a Chord, representing coincidence. Several Chords connected with a partial order form a Phrase.

All patterns are accompanied by linguistic descriptions to better support interpretation. The hierarchical structure inherited from the UTG offers unique possibilities in relevance feedback during the knowledge discovery process and in the analysis of the results. The expert can zoom into the patterns found, and guide the mining process by filtering out irrelevant or uninteresting information at lower levels.

The names of the different constructs of the TSKR are taken from musicology. This was done to avoid confusion with existing definitions in the literature. The Chords are an extension of the Events in the UTG. While Events in the UTG represent intervals, many other publications refer to events as temporal elements occuring at time points without duration (e.g. Jensen et al. (1994); Mannila et al. (1997); Bettini et al. (1998); Guralnik and Srivastava (1999); Weiss (1999); Povinelli (2001); Cotofrei and Stoffel (2005)). The Phrase patterns represent a partial order of intervals. This is very similar to Episodes (see Chapter 6.1), describing a partial order of time points. Nevertheless we decided to use a new term, because the word Episode has also been used in different contexts for interval patterns (Höppner, 2002a), primitive shapes in time series (Colomer et al., 2002), and meaningful short subsequences of symbolic time series (Cohen and Adams, 2001).

7.3 Analysis

Having introduced a new language for the description of interval patterns, we discuss the relation of the TSKR to Allen's interval relations and the UTG. We show that the pattern language of the TSKR has a higher temporal expressivity than the other approaches. We introduce a concept of robustness for symbolic interval patterns and use it to evaluate the pattern languages. Different aspects of interpretability are described according to the Gricean conversational maxims (Grice, 1989; Sripada et al., 2003b). The comparison concludes with a consideration of the availability and feasibility of algorithms for mining of patterns in the different representations.

7.3.1 Expressivity

The interval relations of Allen are widely used in temporal data mining (see Chapter 3.2.2). They can be used to solve the constraint satisfaction problem in temporal reasoning (see Chapter 4.7). But this problem does not occur in the data mining context: the interval data is usually given (Höppner, 2003). As more important for mining patterns from data, we consider the temporal concepts (see Chapter 3) expressible by the relations. Interval rules based on Allen's relations can express the concepts or order, coincidence, and synchronicity. We show that the UTG is a weaker representation, while the TSKR is slightly richer. We use the notation $\subset$ ($\subseteq$) for smaller (smaller or equal) temporal expressivity.

Theorem 7.30. UTG $\subset$ Allen, i.e., the UTG cannot express all patterns expressible with Allen's interval relations.

Proof: By counter-example: Consider the pattern in Figure 7.16 that can be described using Allen's relations, e.g., as *(A equals B) overlaps (C equals D)*. It cannot, however, be expressed by a combination of Events and Sequences of the UTG. The leading part of the first interval pair and the trailing part of the second interval pair are too long for this combination of four intervals to be considered synchronous under all sensible choices of the threshold parameters (Guimarães (1998), p. 41, Definition 3.1.6). All UTG patterns, however, can easily be expressed

with Allen's relations. For intervals within an Event the pairwise relations of the intervals can be *overlaps*, *starts*, *during*, *ends*, the corresponding inverse relations, or *equals*. For intervals from different Events within a Sequence, the relations are *before*, *meets*, or the corresponding inverses. □

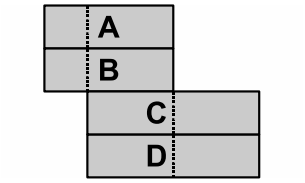


Figure 7.16: Example of interval pattern expressible with Allen's relations but not with the UTG. The intervals cannot be considered almost synchronous.

What is missing in the UTG is the temporal concept of coincidence. Changing the semantic of Event patterns from (almost) synchronicity to coincidence was proposed in (Mörchen and Ultsch, 2005a). Only the center region between the dashed lines in Figure 7.1 was considered for Event patterns. This does not completely alleviate the problem of lower expressivity. An Event was still required to involve intervals from all or all but one of the dimensions (Aspects) describing the problem (Guimarães (1998), p. 42, Definition 3.1.7). The possible leading pattern *A coincides with B* is not allowed in the UTG because it only involves two of the four dimensions of the time series. The Chord patterns of the TSKR not only relax the strict concept of synchronicity to coincidence but further allow groupings of an arbitrary number of intervals. This way the pattern from Figure 7.16 can be represented by a succession of three Chords in a Phrase: *(A,B coincide) then (A,B,C,D coincide) then (C,D coincide).*

Theorem 7.31. Allen $\subseteq$ TSKR, i.e., all patterns expressible with Allen's interval relations can also be expressed with the TSKR.

Proof: By construction: Let $\mathcal{I} = \{(\sigma_i, s_i, e_i) | i = 1, ..., k\}$ be all involved intervals with any pairwise relationship according to Allen. Let $B = \{b_j\} = \bigcup_{i=1}^{k} \{s_i, e_i\}$ the set of all interval boundaries. Construct $|B|-1$ Chords where the j-th Chord describes the coincidence of all Tone instances $(\sigma_i,\ s_i,\ e_i)$ where $s_i \leq b_j$ and $e_i \leq b_{j+1}$. Empty Chords are removed, the remaining Chords form a Phrase with the ordering according to the indices j. □

Example 7.12. The pattern shown in Figure 7.17 consist of six intervals and at least one representative of each of Allen's relations: *A before E*, *B meets E*, *A overlaps B*, *B starts C*, *E during F*, *E finishes C*, and *A equals D*. The five resulting Chords are shown in the bottom row, the Phrase pattern reads *AD then ABCD then BC then BCF then ECF then F* where *AD* is the symbol for the Chord *A and D coincide* and the other Chords are labeled accordingly.

Theorem 7.32. Allen $\subset$ TSKR, i.e., Allen's pairwise interval relations cannot express all patterns expressible with the TSKR.

Proof: By counter-example: Consider the Chords *AB*, *ABC*, *BC*, and *AC* as shown in Figure 7.18(a) at the bottom resulting from the Tone patterns *A*, *B*, *C* in the top rows. In Figure 7.18(b) the same Tones intervals result in the same Chords but with the middle two Chords exchanged. Both versions can be captured in a single Phrase (see Figure 7.18(c)) using the concept of partial order. The order relation of *ABC* and *BC* is simply not specified, both

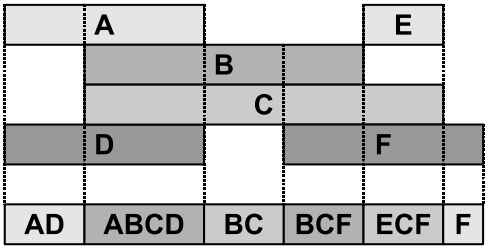
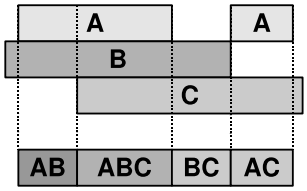
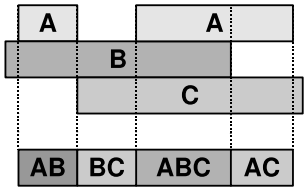


Figure 7.17: Example for the conversion of a complex pattern containing all of Allen's relations to the TSKR pattern language. Whenever an interval starts or ends, a new Chords is created, the sequence of Chords forms a Phrase.

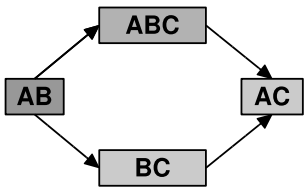
versions of the pattern match this description. Patterns using Allen's pairwise interval relations cannot merge these patterns, because the relation of the first A interval to C changes from *overlaps* to *meets* and the relation of B to the second A from *meets* to *overlapped by*. □



(a) Phrase instance.



(b) Slightly different Phrase instance.



(c) Partial order of Chords within Phrase.

Figure 7.18: Example of a Phrase with partial order among the Chords. The order of the two middle Chords is unspecified. This is not expressible with common pattern formats using Allen's relations.

Corollary 7.33. UTG ⊂ TSKR, i.e., the UTG cannot express all patterns expressible with the TSKR.

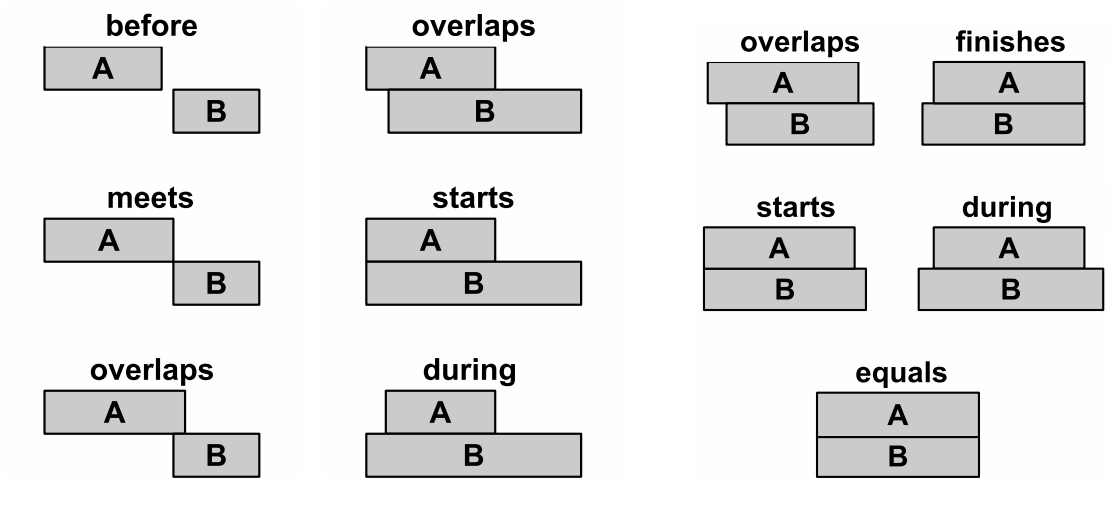
Proof: By transitivity: UTG ⊂ Allen ⊂ TSKR □

7.3.2 Robustness

Symbolic interval series or sequences obtained from numeric time series inherit the noise present in the original data. The interval boundaries gained from preprocessing steps like discretization or segmentation should not be considered exact, but rather approximate. Different parameter settings in the discretization process can strongly influence interval boundaries. In (Höppner, 2002c) the authors mine Allen's relations on weather data obtained from air pressure and wind sensors even though they note that *"temporal processes are often subject to dilation in time"*. We discuss the implications of such noise present in the input data with examples and more formally by introducing a measure of robustness for interval patterns.

The use of Allen's relations is problematic with noisy interval data. Slight variations of interval boundaries can create fragmented patterns that intuitively describe the same relationship between intervals with different operators of Allen's relations. In Figure 7.19 we give several

examples. If two intervals always appear consecutively but close in time, three different relations can be observed depending on the exact location of the first end point and the second start point (see Figure 7.19(a)). Intuitively, the relation between the two intervals common to all three examples could be called *largely before* or *roughly meets*. Similar examples are shown for a pattern where the first interval starts almost synchronously with the second but ends earlier (*roughly starts*, Figure 7.19(b)). The case of *almost equal* intervals is fragmented into nine different patterns of Allen's relations, the five relations shown in Figure 7.19(c) and the corresponding inverses for the first four.



(a) largely before (b) roughly starts (c) almost equals

Figure 7.19: Three examples where intuitively similar situations are fragmented into different relations of Allen due to slightly shifted interval boundaries.

There are approaches to relax the strictness of Allen's relations by using a threshold to consider temporally close interval endpoints equal (Aiello et al. (2002), see Chapter 3.2.2). This effectively merges the problematic patterns in each of the cases shown in Figure 7.19, but it has not been used in temporal pattern mining. Fragmented patterns are still possible if noise causes interval boundaries to be shifted around the threshold value.

The temporal operators of the UTG also use thresholds: the Event patterns include limits on the distances of the interval boundaries to determine whether intervals are more or less simultaneous. The Sequence patterns of consecutive Events allow gaps up to a maximal length between the Events. The final Temporal Patterns use disjunctions to allow variations in the Sequence patterns. This way the UTG performs robust handling of noisy interval boundaries by construction. Within the Sequences, however, a distinction between the *followed by* and *immediately followed by* relations is made (Guimarães (1998), p. 45). Similar to Allen's relations this requires the equality of several interval endpoints and is problematic with noisy data.

In contrast to Allen' relations and the UTG, the Chord and Phrase patterns of the TSKR are designed to be insensitive to small changes of the boundaries of participating intervals. To measure this robustness numerically, we introduce novel measures for the robustness of interval operators. The robustness of statistical estimators is commonly described with the breakdown value, i.e., the minimum fraction of the observations, that make the estimator infinite when set to infinity. We define a similar notion of robustness for interval patterns to compare the rule operators of Allen's relations, the UTG, and the TSKR. For ease of demonstration we will only consider simple patterns relating two symbolic intervals.

Definition 7.34. A pattern relating symbolic intervals has a positive breakdown value b^+ if it can be destroyed by enlarging one interval by at least b^+ ticks and a negative breakdown value b^- if it can be destroyed by making one interval smaller by at least b^- ticks.

In Table 7.2 we list the breakdown values for the three interval rule languages. All of Allen's relations that require equality of at least two interval boundaries can be destroyed by changing one boundary by one tick only. It does not matter whether the intervals get larger or smaller, if the equality is destroyed, so is the pattern. They achieve the lowest possible breakdown values of 1 for both stretching and shrinking intervals. For *before*, *overlaps*, and *during* the amount of change depends on the participating intervals. The further the intervals related with *before* are apart, the larger the breakdown values b^+ is. When shrinking the intervals the *before* patterns does not break down until one of the interval disappears completely. Thus, b^- is simply the shorter length of the two intervals, the largest possible value. The breakdown value b^+ for *overlaps* and *during* depends on the length of the leading and trailing parts where only one interval is present. The longer these parts are, the more robust the pattern w.r.t. stretching. When making the intervals smaller, the *overlaps* operator breaks down if there are no more overlapping parts and *during* breaks down as soon as the first, smaller interval disappears.

	Interval operator	b^+	b^-
Allen	meets	1	1
	starts	1	1
	finishes	1	1
	equals	1	1
	before	$s_2 - e_1 - 1$	$min(e_1 - s_1 + 1, e_2 - s_2 + 1)$
	overlaps	$min(s_2 - s_1, e_2 - e_1)$	$e_1 - s_2 + 1$
	during	$min(s_1 - s_2, e_2 - e_1)$	$e_1 - s_1 + 1$
UTG	more or less simultaneous	$1 \leq b^+ \leq \epsilon$	$1 \leq b^- \leq \epsilon$
	followed by	$1 \leq b^+ \leq \epsilon$	$1 \leq b^- \leq \epsilon$
	immediately followed by	1	1
TSKR	coincides	∞	$min(e_1 - s_1 + 1, e_2 - s_2 + 1)$
	then	∞	$min(e_1 - s_1 + 1, e_2 - s_2 + 1)$

Table 7.2: Breakdown values for Allen's relations, the UTG, and the TSKR operators measuring the robustness to noise in interval boundaries. Only patterns with two intervals $[s_1, e_1]$ and $[s_2, e_2]$ are considered. The threshold parameter for UTG Events is ϵ.

For the *more or less simultaneous* operator from the UTG the amount of change needed to break down the Events pattern depends on how close the maximum difference m among all participating start points (or end points) is. If m is already equal to the threshold parameter ϵ, constraining this maximum difference, a change by one tick will destroy the Event pattern. If all intervals are completely simultaneous with all start and end points being equal, respectively, the change needs to be as large as ϵ. This is true for both shrinking and expanding intervals. The *followed by* operator of UTG Sequence patterns directly depends on the Event patterns. It breaks down as soon as one of the participating Events breaks down. The *immediately followed by* operator requires several endpoints to be equal and thus has the worst breakdown values possible, similar to several of Allen's relations.

The TSKR operator *coincides* is extremely robust when making intervals larger. Since it only considers the intersection of all participating intervals, any interval can be stretched to infinity without changing the pattern at all. When making intervals smaller, the pattern breaks down as soon as the smallest interval disappears. Similar to the UTG, the *then* operator depends on the *coincides* operator.

In summary, we can see that the TSKR patterns have excellent robustness properties according to our definition of pattern breakdown. The *coincides* operator is only concerned about the time interval where it is valid. Any change to the participating symbolic intervals outside of the occurrence interval is irrelevant. The breakdown when making intervals smaller also achieves the largest possible value. The UTG patterns are mostly robust, the amount of difference in the interval boundaries is directly controlled by the threshold parameter for Events. Seven of the 13 relation of Allen have the worst breakdown values for both enlarging and shortening intervals.

7.3.3 Interpretability

Interpretability is hard to specify or measure. We are not trying to propose a general definition of interpretability. Rather we want to look at different aspects of the pattern languages that influence how easily the patterns can be comprehended and interpreted by humans. (Sripada et al., 2003b) suggests to use the Gricean conversational maxims (Grice, 1989) of quality, relevance, manner, and quantity when communicating data mining results to experts for analysis. We therefore describe the implications of the maxims in the context of interval patterns and judge how well the different pattern languages support this paradigm.

The maxim of quality states that only well supported facts and no false descriptions should be reported. The former is commonly achieved by reporting only the most frequent patterns. None of the pattern languages can be said to report false patterns. As described in Chapter 7.3.2, noisy interval data can lead to fragmented patterns expressed with Allen's relations, however, that might not be reported even if they are frequent.

The maxim of relevance requires, that only patterns relevant to the expert are listed. This is a rather application dependent requirement, still there are differences in how the pattern languages support this task. The hierarchical structure of the UTG and TSKR enables the user to view and filter lower level patterns before the next level constructs are searched. E.g. mining Phrases from only the relevant Chords will be much faster, fewer and smaller Phrase patterns need to be analyzed in the next step. The patterns expressed with Allen's relations are mined in a single step. The possibly much larger set of patterns needs to be checked for relevance in a single post-processing step.

The maxim of manner requires communication to be brief and orderly and to avoid obscurity and ambiguity. The patterns of the UTG and TSKR are generally briefer than Allen patterns. The hierarchical form hides the complexity of lower levels achieving a concise high level representation. An expert can zoom into each rule to learn about how it is composed and what its meaning and consequences might be. This form of pattern representation supports the human analysis according to the zoom, filter, and details on demand paradigm (Shneiderman, 1996).

The UTG and TSKR also have an advantage in the number of operators needed for k intervals. All of Allen's relations are binary operators. A single pattern with k intervals consists of $\frac{k(k-1)}{2}$ pairwise relations. A textual listing of these single relations is hard to grasp. In Figure 7.20 we give an example of a pattern with six intervals that is expressible by Allen's relations, the UTG, and the TSKR. The pattern is described with 15 relations according to Allen in Figure 7.21. It could be made slightly more compact by grouping the intervals A,B,C and D,E,F, respectively and relating the groups with *before*. Such a grouping is not supported by (Höppner, 2002c; Kam and Fu, 2000), however, and not explicitly sought in (Cohen et al., 2002). The UTG and the TSKR descriptions of the pattern in Figure 7.20 are very similar for this particular pattern (see Figure 7.22 and Figure 7.23, respectively). The Event and Chord patterns group several intervals with a single operator, making the final pattern more compact.

The pairwise relations of patterns expressed with Allen's relations can be ordered using the order of the intervals according to start and end points. The Sequence and Phrase patterns of the UTG and TSKR explicitly describe an ordering of the lower level constructs.

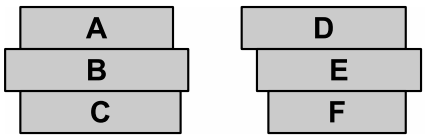


Figure 7.20: Example of a pattern expressible by Allen, UTG, and TSKR.

A *during* B	B *contains* C	C *before D*	D *overlaps* E	E *contains* F
A *starts* C	B *before* D	C *before E*	D *finished* by F	
A *before* D	B *before* E	C *before F*		
A *before* E	B *before* F			
A *before* F				

Figure 7.21: The textual representation of Figure 7.20 using Allen's pairwise relations.

To avoid ambiguity, a fixed pattern description should semantically describe a single intuitive notion of the relationships of several intervals with each other. This is not the case for all of Allen's relation. The instances of a single pattern can vary significantly and represent intuitively different patterns. This was already demonstrated for the *overlaps* relation in Figure 7.3, two more examples for the relations *starts* and *during* are given in Figure 7.24. The different instances that are described by the same relation appear visually quite distinct. Whether these different versions of a pattern are semantically equivalent depends on the application.

This high variation of patterns represented by a single relation of Allen is caused by the mixture of the temporal concepts of order and coincidence. The problem can be weakened by using the Allen's relations modified with thresholds (see Chapter 3.2.2), but the potential ambiguity persist, in particular for intervals of very different lengths.

In contrast to Allen's relations, the UTG and TSKR clearly separate the temporal concepts within the rules. Event patterns only describe almost equal intervals and the variation of the instances is controlled by the threshold parameter. Chord patterns simply describe the overlapping parts and ignore any variability of the interval boundaries. The parts of the Tone intervals within the Chord occurrence interval always look exactly the same for all instances. Sequence patterns express the concept of order, allowing variation only in the length of Events and possible gaps. Similarly, Phrases handle partial order.

The ambiguity problem is even worse for the pattern format used in (Kam and Fu, 2000) and (Cohen, 2001). Recall that while (Höppner, 2001) represents patterns by listing all pairwise relations between participating intervals, the other two methods successively add new intervals to a pattern by expressing the relation of the intervals to the complete previous pattern. A more compact representation with only $k-1$ relations for k intervals is achieved. The problem is, that the *exact same* instance of a pattern can be represented by many different compact formulas. An example is given in Figure 7.25. To be precise, a pattern with k intervals can be written in $\frac{k!}{2}$ ways by consecutively choosing an interval for the next position in the pattern and excluding duplicates caused by inverse operators.

The maxim of quantity states that neither too much nor too few should be reported. Fewer and shorter patterns are easier to read and comprehend. Achieving this maxim is again better supported by the hierarchical languages. As described above the patterns are generally briefer. Intermediate filtering steps during the pattern mining can be used to prune the search space for the next step and reduce the number of patterns to be inspected. The margin-closedness of the TSKR explicitly offers a powerful mechanism to reduce the number of reported patterns without sacrificing variety. Implication rules generated from Allen patterns can be ranked by interestingness measures (Höppner and Klawonn, 2002), but for the basic pattern considered

Sequence S ← ABC *before* DEF.
Event ABC ← A, B, and C are *more or less simultaneous.*
Event DEF ← D, E, and F are *more or less simultaneous.*

Figure 7.22: The textual representation of Figure 7.20 using the UTG.

Phrase S *when* ABC *then* DEF.
Chord ABC *when* A, B and C *coincide.*
Chord DEF *when* D, E and F *coincide.*

Figure 7.23: The textual representation of Figure 7.20 using the TSKR.

here, only the frequency can be used. It too many patterns are reported the frequency threshold needs to be raised possibly filtering out rare but interesting patterns.

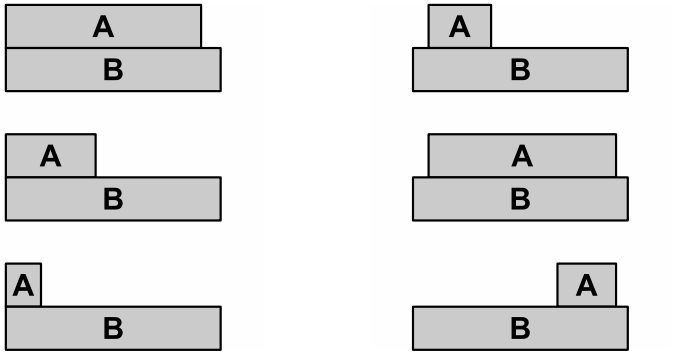
In summary, the UTG and especially the TSKR show more accordance with the Gricean maxims than Allen's relations. The maxim of quality can be violated by fragmented Allen patterns. The maxim of relevance is better supported by the interactive filtering of patterns possible for UTG and TSKR. For the maxim of manner brevity of representation is achieved by the hierarchical structure of UTG and TSKR with operators grouping several intervals as opposed to the pairwise listing of binary relations from Allen. The restriction to single temporal concepts for each level of the hierarchical languages avoids ambiguity. The TSKR best supports the maxim of quantity by effective pruning of the results using margin-closedness.

7.3.4 Algorithms

Höppner describes a mining algorithm for Allen's relations based on the Apriori principle from association rules (Höppner, 2001). The patterns are build in a breadth first manner creating patterns with k intervals from two patterns with $k-1$ intervals, $k-2$ of which are equal. It is well known, however, that this breadth first approach is inferior in performance to depth-first search for long patterns (Wang and Han, 2004). Some authors criticize the concept of candidate generation and subsequent support estimation for association rules and sequential patterns in general (Han and Pei, 2001; Pei et al., 2001). Transferring the concepts of closedness and maximality w.r.t. support (e.g. Pei et al. (2004)) to interval patterns expressed with Allen's relations might be a solution to reduce the number of patterns found. In any case, due to the high expressivity of patterns expressed with Allen's pairwise relations, the search space for such patterns is large, making pruning techniques like using a minimum support threshold necessary.

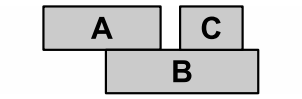
Guimarães and Ultsch do not describe any algorithms to mine the Temporal Patterns of the UTG, most processing steps are performed manually (Guimarães and Ultsch, 1999). This is unfortunate, because the separation of temporal concepts will make the pattern spaces smaller than when mining Allen patterns in one step. In a first attempt to automate the discovery of UTG patterns, we have proposed two novel and several known algorithms for mining the various temporal relations corresponding to the UTG hierarchy (Mörchen and Ultsch, 2005a). Event patterns can be mined with a simple linear time algorithm, for Sequence patterns a suffix trie method (Vilo, 1998) was used.

The algorithms for mining Events cannot be easily transferred to Chords, because in contrast to Events the Chord patterns can overlap. The suffix trie method can only mine total order and cannot be used to mine Phrases. We will describe efficient algorithms for mining TSKR patterns in Chapter 8, completing the TSKM methodology.



(a) starts (b) during

Figure 7.24: Three different instances of Allen's *starts* and *during* relation that have quite distinct appearances.



(a) Pattern with three intervals

(A *overlaps* B) *contains* C
(A *before* C) *overlaps* B
(B *contains* C) *overlapped by* A

(b) Different representations of the same pattern.

Figure 7.25: Example for the ambiguity of patterns represented with Allen's relations in the compact format of (Kam and Fu, 2000; Cohen, 2001). The same pattern can be described with three different rules.

7.3.5 Summary

The summary of this analysis of interval pattern languages is listed in Table 7.3. Both Allen and the UTG have major disadvantages. The main problem of the UTG is the lower expressivity caused by restricting the number and alignment of the intervals in Event patterns. Many simple Allen patterns are not expressible with the UTG. The TSKR has been proposed as a solution, resulting in the highest temporal expressivity of the three languages. The missing robustness of Allen's relations can be compensated by introducing thresholds for the matching of interval boundaries. This does not solve the problem of ambiguity, however. We think that the interpretability of Allen is inferior because the pattern are ambiguous and potentially much longer. The hierarchical structure of UTG and TSKR with consistent use of semiotic triples creates shorter and more abstract patterns and provides early pruning and details on demand. Ambiguity is avoided by the separation of temporal concepts.

Comparison	Allen	UTG	TSKR
Expressivity	+	−	++
Robustness	(−)	+	++
Interpretability	−	+	+
Algorithms	+	(−)	+

Table 7.3: Summary of the comparison of Allen, UTG, and TSKR pattern languages for mining temporal rules from time interval data.

Chapter 8

Algorithms for Mining Time Series Knowledge

This chapter describes algorithms for finding temporal knowledge in multivariate time series expressed with the TSKR. This completes our proposed method for Time Series Knowledge Mining (TSKM). We define the mining tasks for each level of the TSKR hierarchy and describe algorithms to solve these tasks. There are basically five processing steps in the TSKM shown in Figure 8.1. Various methods for preprocessing are briefly explained in Chapter 8.1. The benefits of grouping the dimensions into Aspects are highlighted in Chapter 8.2. For mining Tones, many different algorithms can be used, we list some examples of existing methods in Chapter 8.3. The main part of this Chapter follows in the description of novel algorithms for the efficient discovery of Chords and Phrases in Chapter 8.4 and Chapter 8.5, respectively.

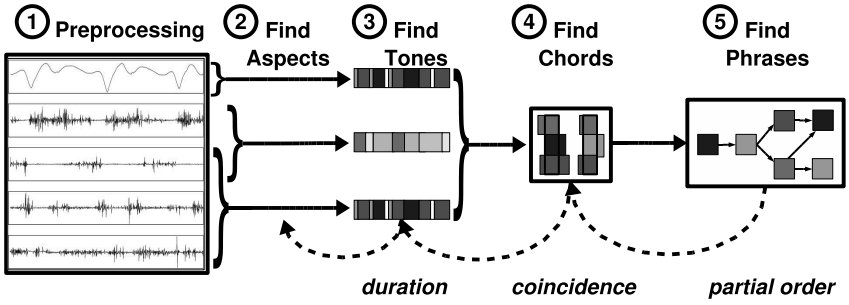


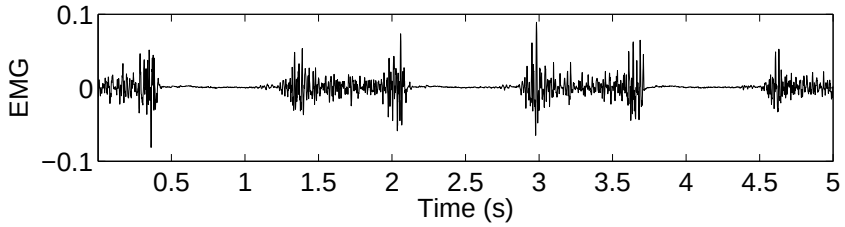
Figure 8.1: The five main steps for the discovery of TSKR patterns are depicted. After preprocessing and Aspect grouping, the three pattern classes Tones, Chords, and Phrases are subsequently mined. The dotted arrows represent revisions based on intermediate results.

As with knowledge discovery in general (see Chapter 2.1) iterations of previously executed steps are often advisable. This is indicated by the dashed arrows at the bottom of Figure 8.1. After mining patterns for one level of the TSKR hierarchy, a careful examination of the results should be performed. Intermediate results can give insights into the data and temporal structures and might suggest a revision of previously set parameters. Filtering out spurious or irrelevant results helps do reduce the search space for patterns on the next level. Mining the TSKR patterns involves not only finding the occurrences of the patterns, but also a symbol and label to complete the semiotic triples. The unique symbols can easily be generated automatically during the mining process. The labels can also be set automatically, but using meaningful names, ideally chosen by a domain expert, will improve the interpretability of the higher levels patterns.

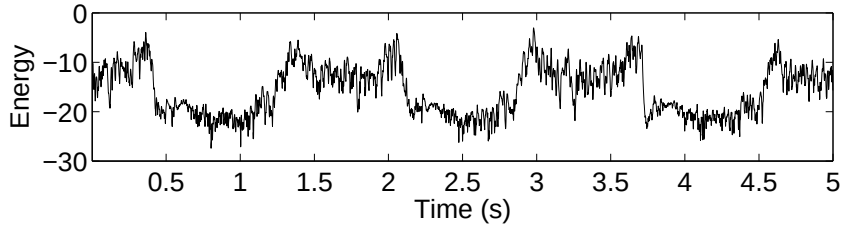
8.1 Preprocessing

Preprocessing and feature extraction techniques can be applied to the complete multivariate time series or to each variable separately. This is a highly application dependent step, see Chapter 5.2 for a description of common methods for the removal of noise and outliers or the extraction of additional features. We give a real life example here to demonstrate the importance and complexity of this step.

Example 8.1. The EMG[1] signals of the Skating data (Chapter A.1) were measured with surface electrodes attached to the skin above the major leg extensor muscles. The electrodes capture small currents induced by the action potentials of different muscle fibers. An example signal of five seconds is shown in Figure 8.2(a). The interesting information of the EMG signals is hidden in the amplitude and frequency of the activation current. The frequency is related to type and function of the muscle fiber and its corresponding motor unit. Fast-twitch, fast fatiguing muscle fibers emit higher frequencies, while slow-twitch, slow fatiguing muscles contribute lower frequency components to the EMG signal. The amplitude is related to the intensity of activation and the number of muscles fibers being activated. An analysis of the time-frequency distribution was performed using the Morlet CWT (Grossmann and Morlet, 1984) in the relevant frequency range for EMG signals from 10Hz to 250Hz. The instantaneous energy was calculated as the logarithm of the sum of all wavelet coefficients for a time point. The resulting time series represents the muscle activation and was used for further processing. The excerpt for the example is shown in Figure 8.2(b).



(a) EMG signal



(b) Muscle activation

Figure 8.2: Preprocessing of EMG signal from the Skating data to derive instantaneous muscle activation.

[1] Electromyography

8.2 Aspects

Task 8.1. Mining Aspects

Description	The tasks of finding k Aspects of a d-dimensional multivariate time series Y is the selection of k subsets of the dimensions $\{1, ..., d\}$ with an appropriate labeling.
Input	d-dimensional time series Y.
Output	k Aspects $A_i \subset Y, i = 1, ..., k$.

Following the divide-and-conquer paradigm it is often advisable to group the dimensions of a multivariate time series into semantical blocks, the Aspects. This tasks will usually be performed manually using a priori knowledge about the domain. In lack of any information about relations of the different dimensions, a trivial grouping into d univariate Aspects can be performed. The subsets of the dimensions are allowed to be overlapping. Note, that this can introduce undesired correlation between the Aspects.

After mining the Aspects of the input time series, another iteration of preprocessing should be considered. Multivariate Aspects in particular can profit from further processing. In order to prepare a multivariate Aspects for finding Tones by clustering, normalization of the mean values and variances should be performed to avoid undesired emphasis of some variables within the Aspect. The dimensionality of an Aspect consisting of highly correlated variables could be reduced using PCA (Jolliffe, 1986), possibly even down to a univariate Aspect. Independent influences can be revealed using ICA Hyvärinen (1999).

Example 8.2. In the Skating data (Chapter A.1) the movements of the skater's leg are determined with three electrogoniometers measuring the angles of the ankle, knee, and hip joints. There is a functional dependency between these three variables that is not of major interest for the application. We did not want to analyze the relation of, e.g., the knee angle to the hip angle, but rather the activation of muscles during different movement phases. The three angular time series were therefore grouped into an Aspect called *Movement* collectively describing the leg's movement. The movements at each joint are repeatedly observed in two directions: extension and flexion. This was incorporated into the Aspect with three time series measuring the signed angular velocity, derived from the angle values with a differentiation filter. This resulted in a 6-dimensional time series describing the movement of the leg.

8.3 Finding Tones

Task 8.2. Mining Tones

Description	The tasks of finding a Tone for an Aspect is the creation of a Tone and the symbolic interval series with the maximal occurrences thereof.
Input	Aspect A.
Output	Tone t and a symbolic interval series $\mathcal{T}$.

Tone patterns describe persisting properties of an Aspect occurring on time intervals. Since we allowed arbitrary functions describing a Tone in Chapter 7.2.4 there are many algorithms that can be used to find such patterns. We briefly describe several possible methods for mining Tones based on value, trend, or shape.

8.3.1 Univariate value-based Tones

Value-based Tones for univariate time series can be found by partitioning the value axis into bins. We have described various methods for this in Chapter 5.4.3. This results in Tone patterns

for each bin that can easily be labeled with linguistic descriptions like *high* or *low*. The intervals are obtained by concatenating runs of values in the same bin.

Static discretization methods like equal frequency or equal width histograms ignore the temporal order of the values, however. This is likely to produce noisy state intervals. One possibility to incorporate the temporal structure of the values is the training of a HMM and the extraction of the Viterbi state sequence. But the training of HMM (Rabiner, 1989) models is slow and vulnerable to outliers. HMM models are also harder to interpret than the result of binning methods. The PERSIST algorithm is a simple binning method that utilizes the temporal order of values and has been shown to outperform many static methods and HMM in recovering known states (Mörchen and Ultsch, 2005b).

8.3.2 Univariate trend-based Tones

Many time series segmentation methods create a series of intervals with linear functions per interval approximating the local trend of the values (see Chapter 5.8). Each interval is an instance of a Tone pattern according to a discretization of all observed slopes from the functions. Descriptions for the Tones can accordingly be generated as *increasing* or *flat*. Second order methods additionally incorporate the second derivative of the signal to distinguish convex from concave trends (Höppner, 2001).

The Bottom-Up algorithm is a good candidate for producing the segmentation. It runs in linear time, compared to the quadratic time complexity of Top-Down. Since the complete data is available at the time of mining, the more global view of the Bottom-Up compared to the sliding windows approach is advantageous (Keogh et al., 2004a). One practical problem with Bottom-Up (and the other segmentation approaches) is that the number of segments or an error threshold need to be specified a priori. Permutation tests can help in choosing the parameters (Vasko and Toivonen, 2002). An alternative method is the multi-scale analysis described in (Höppner, 2003). The time series is segmented using the most dominant features in the scale space which is similar to how humans would describe time series. The number of segments is determined automatically.

8.3.3 Univariate shape-based Tones

Shape-based Tones represent frequently occuring short sub-series and can be found, e.g., by clustering segments of the time series (see Chapter 5.7.2). Each cluster corresponds to a Tone. The segments can be obtained using a sliding window, but choosing the window size is problematic and overlapping sliding windows produce meaningless results (Lin et al., 2003b). Time series segmentation methods create non-overlapping segments (see Chapter 5.8). Typically the complete time series is segmented. If only the most frequent and typical shapes are of interest, time series motif discovery (see Chapter 5.10) can be used to determine the shapes and their locations simultaneously. If some shapes of interest are known a priori, the problem of finding the Tones occurrences is simply that of retrieval by content (see Chapter 5.12) supporting subsequence queries.

8.3.4 Multivariate value-based Tones

For multivariate time series time point clustering (see Chapter 5.7.3) can be used to find value-based Tones. One Tone is created for each cluster, the cluster membership of the time points determines the occurrence intervals. If the process alternates between several regimes or states, these regions should form clusters in the high dimensional space of the features vectors per time point. Persisting clusters are characterized by long intervals with the same cluster membership, compared to randomly scattered cluster assignments on the time axis.

The cluster algorithm used for time point clustering should be able to cope with outliers and offer a validation of the existence of clusters. One such powerful method is the use of Emergent Self-organizing Maps (ESOM, Ultsch (1999)) with the U-Matrix (Ultsch, 1993) visualization. Outliers are clearly visible on the maps. They can be removed and a new map is trained with the remaining data. The U-Matrix visualizes clusters of arbitrary shape and size. A trained and manually labeled ESOM can be used to classify new data based on the identified cluster structure.

For the description of each Tone a symbolic rule should to be generated. This provides a meaningful basis for the further discovery of temporal rules. A method explicitly designed to describe clusters with interval conditions using only the most relevant variables per cluster is the Sig* algorithm (Ultsch, 1991). Other rule generation methods (see Witten and Frank (1999) for an overview) can be used as well, but they are mostly designed to create discrimination rules that focus on the differences of the clusters.

If there is some training data available on which the assignment of the time points to known states is known, Tones can also be learned with a classifier. Interpretable classifiers like decision trees are to be preferred over methods like SVM (Burges, 1998) in the context of knowledge discovery.

8.3.5 Finding marginally interrupted occurrences

Task 8.3. Mining marginally interrupted Tones

Description	The tasks of finding the marginally interrupted occurrences of several Tones is the filtering of short gaps from the symbolic interval series with the maximal occurrences of the Tones.
Input	Tones T and symbolic interval series $\mathcal{T}$.
Output	Symbolic interval series $\mathcal{T}'$.
Parameters	Maximum relative total length α of all interruptions. Maximum absolute length d_{min} of any interruption.

The instances of any Tone pattern can be interrupted by noise. An occurrence of a value-based Tone can be interrupted by an outlying value, splitting the potentially longer interval in two parts. Similar, for trend-based patterns an outlier during an otherwise flat segment can lead to two surrounding segments with large slopes. In order to filter out such short interruptions from the maximal occurrences of Tones we propose a filtering algorithm that ensures the conditions for marginally interrupted occurrences.

Algorithm 8.1, called MarginalGapFilter, first searches for all pairs of intervals with the same symbol separated by a short enough gap (Lines 1-6). The gaps are sorted in increasing order by duration to process shorter gaps first. If a long interval is interrupted by several short gaps of different lengths and the relative total length of all gaps exceeds the threshold α we rather want to remove many short gaps until the threshold is met, than to remove few large gaps and leave the small gaps in the data. In Line 8 the output interval series is initialized with the input intervals augmented with their respective duration. This is needed because after replacing gaps and creating longer intervals, we still need the information about the original lengths of the merged intervals to prevent more than only marginal interruptions. For all gaps we check in Line 10 whether there still exist two intervals immediately to the left and right of the gap. Further we compare the sum of all durations from the original intervals contained in the left and right intervals to the duration of the long interval obtained by removing the gap. If the threshold condition posed by α is not violated the gap is removed in Line 11 by replacing the two intervals with a single long interval adding up the stored original interval durations.

Algorithm 8.1 MARGINALGAPFILTER finds marginally interrupted occurrences from maximal occurrences.

Input:
Symbolic interval series $\mathcal{T} = \{(s_i, e_i, \sigma_i) | i = 1, ..., n\}$.
Total maximum relative interruption length α, default $\alpha = 0.1$.
Maximum absolute interruption length d_{max}.

Output:
Symbolic interval series $\mathcal{T}'$.

$G := \emptyset$// the set of all short gaps
for $i = 1$ to n **do**
  for all $j < i$ with $\sigma_j = \sigma_i \wedge (s_i - e_j + 1 \leq d_{max})$ **do**
    $G := G \cup \{(e_j, s_i)\}$
  end for
end for
Sort gaps G by increasing duration
$\mathcal{T}' := \mathcal{T}$ extending (s_i, e_i, σ_i) to $(s_i, e_i, \sigma_i, d_i = e_i + s_i + 1)$
for all $(l_i, r_i) \in G$ **do**
  if $\exists (s_1, e_1, \sigma_1, d_1), (s_2, e_2, \sigma_2, d_2) \in \mathcal{T}'$ with $(e_1 = l_i) \wedge (s_2 = r_i) \wedge \left(\frac{d_1 + d_2}{e_2 - s_1 + 1} \geq 1 - \alpha\right)$ **then**
    $\mathcal{T}' := \{(s_1, e_2, \sigma_1, d_1 + d_2)\} \cup \mathcal{T}' \setminus \{(s_1, e_1, \sigma_1, d_1), (s_2, e_2, \sigma_2, d_2)\}$
  end if
end for

The time complexity of the MARGINALGAPFILTER algorithm is $O(n \log n)$ where n is the number of intervals. The search for all short gaps can be implemented with a linear scan of the interval series keeping track of the most recent intervals for each symbol in a hash table. Since only gaps between temporally close intervals with the same symbol are considered, the number of gaps is linear in the number of intervals. The processing of the gaps in the second part of the algorithm is thus also linear. The sorting of the gaps, however, lifts the total time complexity to $O(n \log n)$.

As a default choice $\alpha = 0.1$ can be used, i.e., at most 10% of a symbolic interval are allowed to be disturbed by interruptions. The d_{max} parameter has to be chosen w.r.t. the application. Often, some knowledge on the minimum duration of a phenomena to be considered interesting is available.

Example 8.3. Value based Tones were generated for the muscle activation of the Skating data (see Chapter A.1) with the PERSIST algorithm and three bins corresponding to the states *low*, *high*, and *very high*. A example of 1.3 seconds is shown in the top panel of Figure 8.3. The two bin boundaries are shown as horizontal lines. The upper bar of shaded intervals (labeled *before*) corresponds to the maximal Tones found by Persist. In places where the activation crosses a bin boundary frequently, short intervals are obtained, for example between 3000ms and 3200ms. These short interruptions of the surrounding longer intervals were filtered with the MARGINALGAPFILTER algorithm using a maximum gap length of 50ms. Short bursts of muscle activation below this threshold are physiologically not plausible and likely to be caused by measurement inaccuracy. The obtained marginally interrupted Tones are shown in the lower bar, labeled *after*.

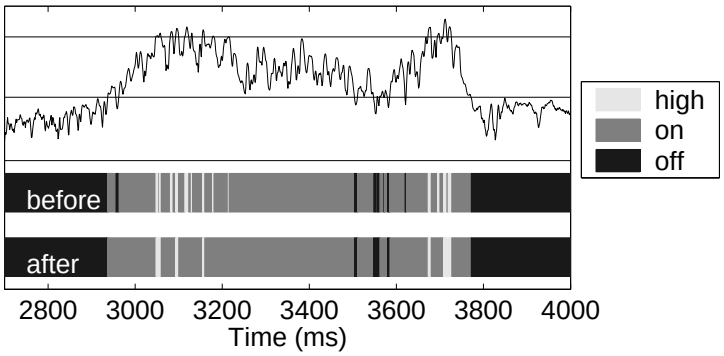


Figure 8.3: The muscle activation of the Skating data is discretized into three states. The maximal Tones are shown in the top row of intervals labeled *before*. After filtering out short gaps, the marginally interrupted Tones in the bottom row labeled *after* are obtained.

8.4 Finding Chords

Task 8.4. Mining Chords

Description	The tasks of finding (margin-closed) Chords in a symbolic interval sequence is the creation of k (margin-closed) Chords and the symbolic interval sequence with the maximal occurrences thereof.
Input	Symbolic interval sequence $\mathcal{T}$ representing occurrences of a set of Tones.
Parameters	Threshold δ for the minimum duration of a Chord. Minimum support of a Chord. Minimum and maximum size of a Chord. Flag whether to mine closed Chords. Threshold α for mining margin-closed Chords.
Output	k Chords C_i and a symbolic interval sequence $\mathcal{C}$.

Chords represent the concept of coincidence and are thus mined from the interval series of Tone occurrences from all Aspects collectively. As described in Chapter 7.2 a Chord is quite similar to an itemset used in association rules mining. We will therefore adapt the depth-first approach of CHARM (Zaki and Hsiao, 2002), an algorithm for mining closed itemsets that has recently been demonstrated to outperform many other competing methods (Zaki and Hsiao, 2005). Another benefit of this method is the possibility to simultaneously construct a lattice structure of the closed itemsets for visualization purposes. We will only describe the core algorithm and refer the interested reader for performance optimizations and lattice construction to (Zaki and Hsiao, 2005).

Let us first point out the differences of the novel task of mining closed Chords compared to mining closed itemsets. The core part of a Chord is the set of Tone symbols that coincide. Only one Tone symbol per Aspect is allowed in a Chord, because a single Aspect cannot be in two states at the same time. This reduces the search space because not all combinations of small Chords form a valid larger Chord. For a Aspects with k Tones each, there are $\binom{a}{s}k^s$ possible Chord patterns of size $s < a$. The support of an itemsets is measured with the number of database transactions in which all items of the set occur. In our setting we can imagine one transaction per time point containing the symbols of all currently valid Tones. Simply counting the points is not enough however, we sum up the lengths of all occurrence intervals that meet the minimum duration threshold δ. Finally, the concept of margin-closedness needs to be incorporated in the algorithm to further reduce the number of patterns found.

Algorithm 8.2, called ClosedChordMining, searches for margin-closed Chords. The search tree defined by the partial order on Chords is explored in a depth-first manner. The function EXTEND is recursively called given a prefix Chord c_p and a set of super-Chords S. The recursion is started with an empty Chord and all trivial frequent Chords as possible extensions in Line 1. The prefix Chord c_p is extended by all combinations of super-Chords from the set S in the double loop starting in Line 3. The variable $\hat{c}_i$ (Line 4) stores the initial super-Chord extension by c_i. The extensions are filtered for valid Chords with only one Tone per Aspect and by the minimum support in Line 7. The core part of the algorithm is the comparison of the support of the super-Chord $\hat{c}_i \cup c_j$ with the support of both sub-Chords $\hat{c}_i$ and c_j. The following four cases (Lines 8, 11, 13, and 16) are generalizations of the four properties for itemsets in (Zaki and Hsiao, 2005) to margin-closed Chords and will be explained in detail below. In Line 22 the current Chord $\hat{c}_i$ is added to the set of closed Chords if it is not subsumed by an already found closed Chord. The function EXTEND is called recursively in Line 25 unless the maximum Chord size is reached. The recursion stops as soon as the set of extensions S is empty.

The four different cases of the support comparison are depicted for an example in Figure 8.4. Let A, B, C be some Tone patterns. Only one instance of each Tone pattern is shown for demonstration purposes, the length and overlaps of the intervals are meant to be representative for the whole data set. For our example, let the arguments to the EXTEND function be the trivial Chord $c_p = A$ and the set of super-Chords be $S = \{AB, AC\}$. For the first extension $\hat{c}_i = A \cup AB = AB$ the support of $\hat{c}_i \cup c_j = AB \cup AC = ABC$ needs to be compared to the support of $\hat{c}_i = AB$ and $c_j = AC$.

The first case in Line 8 is shown in Figure 8.4(a). The super-Chord ABC has almost the same support as both sub-Chords. When AB or AC are observed, so is almost always ABC. We can effectively replace both sub-Chords with the super-Chord by adding $c_j = AC$ to the current extension $\hat{c}_i$ and excluding it from further processing in the inner loop.

The second case in Line 11 is shown in Figure 8.4(b). While AB has the same support as ABC, AC has significantly more. Again we can add AC to the current extension $\hat{c}_i$ because whenever AB occurs, so does AC. We can not delete AC from E in this case, however, because combinations with other Chords still need to be considered.

The third case in Line 13 is shown in Figure 8.4(c). The Chord AB has significantly more support than ABC, while AC does not. The super-Chord ABC is possibly a closed Chord, it needs to be added to the set of immediate super-Chords of $\hat{c}_i = AB$ for the recursive function call. The Chord AC can be excluded from further processing, because whenever it is observed, so is AB.

The last case in Line 16 is shown in Figure 8.4(d). Both AB and AC have significantly more support than ABC. ABC is added to the set of possibly closed super-Chords for the recursion and no pruning can be performed.

The time complexity of Algorithm 8.2 is hard to analyze due to the recursion and the dynamic content of S determining the iterations of the double loop. In (Zaki and Hsiao, 2005) the CHARM algorithm is reported to scale up linearly with the number of transactions. We expect similar behavior with ClosedChordMining, because the number of possible Chord patterns does not depend on the number of symbolic Tone intervals, but rather on the number of different Tone patterns.

The parameter δ needs to be chosen according to the application domain. The minimum duration of a Chord needs to be long enough for an expert to consider the coincidence of several properties meaningful. Similar, information on useful minimum and possibly maximum size constraints of a Chord can be obtained from an expert. The maximum size is of course always bounded by the number of Aspects. The minimum support and the threshold for margin-closedness directly control the number of found Chords. Just like for Tones, the maximal occurrences of Chords can be filtered for gaps to create marginally interrupted occurrences

Algorithm 8.2 ClosedChordMining finds margin-closed Chords by depth-first search.

Input:

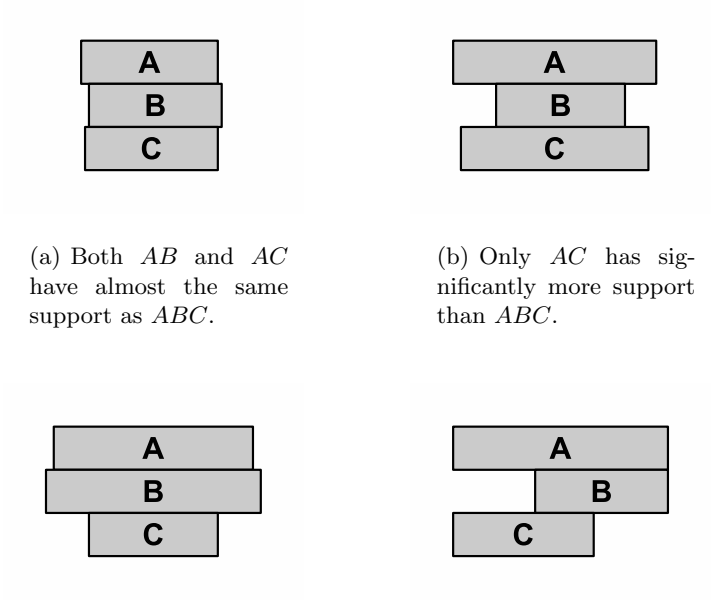
Symbolic interval sequence $\mathcal{T}$.
Minimum Chord duration δ.
Minimum Chord support sup_{min}, default $sup_{min} = 0.01$.
Minimum Chord size s_{min}, default $s_{min} = 2$.
Maximum Chord size s_{max}, default $s_{max} = a$.
Threshold α determining margin-closedness, default $\alpha = 0.1$.

Output:

Set of Chords and symbolic interval sequence $\mathcal{C}$.

1: $S := \{t \in \mathcal{T} | sup(t) >= sup_{min}\}$
2: $R :=$ EXTEND$(\emptyset, S, \emptyset)$

EXTEND(c_p, S, R)
3: **for all** $c_i \in S$ **do**
4: $\quad \hat{c}_i := c_p \cup c_i$
5: $\quad \hat{S} := \emptyset$
6: $\quad$ **for all** $c_j \in S$ with $j > i$ **do**
7: $\quad\quad$ **if** $\hat{c}_i \cup c_j$ is a valid Chord and $sup_\delta(\hat{c}_i \cup c_j) \geq sup_{min}$ **then**
8: $\quad\quad\quad$ **if** $\frac{sup_\delta(\hat{c}_i \cup c_j)}{max(sup_\delta(\hat{c}_i), sup_\delta(c_j))} \geq 1 - \alpha$ **then**
9: $\quad\quad\quad\quad \hat{c}_i := \hat{c}_i \cup c_j$
10: $\quad\quad\quad\quad S := S \setminus \{c_j\}$
11: $\quad\quad\quad$ **else if** $\frac{sup_\delta(\hat{c}_i \cup c_j)}{sup_\delta(\hat{c}_i)} \geq 1 - \alpha$ and $\frac{sup_\delta(\hat{c}_i \cup c_j)}{sup_\delta(c_j)} < 1 - \alpha$ **then**
12: $\quad\quad\quad\quad \hat{c}_i := \hat{c}_i \cup c_j$
13: $\quad\quad\quad$ **else if** $\frac{sup_\delta(\hat{c}_i \cup c_j)}{sup_\delta(\hat{c}_i)} < 1 - \alpha$ and $\frac{sup_\delta(\hat{c}_i \cup c_j)}{sup_\delta(c_j)} \geq 1 - \alpha$ **then**
14: $\quad\quad\quad\quad \hat{S} := \hat{S} \cup \{\hat{c}_i \cup c_j\}$
15: $\quad\quad\quad\quad S := S \setminus \{c_j\}$
16: $\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad \hat{S} := \hat{S} \cup \{\hat{c}_i \cup c_j\}$
18: $\quad\quad\quad$ **end if**
19: $\quad\quad$ **end if**
20: $\quad$ **end for**
21: $\quad$ **if** $|\hat{c}_i| \geq s_{min}$ and $\forall c \in R$ with $\hat{c}_i \subseteq c$ holds $\frac{sup_\delta(c)}{sup_\delta(\hat{c}_i)} < 1 - \alpha$ **then**
22: $\quad\quad R := R \cup \{\hat{c}_i\}$
23: $\quad$ **end if**
24: $\quad$ **if** $|\hat{c}_i| < s_{max}$ **then**
25: $\quad\quad R :=$ EXTEND$(\hat{c}_i, \hat{S}, R)$
26: $\quad$ **end if**
27: **end for**



(a) Both AB and AC have almost the same support as ABC.

(b) Only AC has significantly more support than ABC.

(c) Only AB has significantly more support than ABC.

(d) Both AB and AC have significantly more support than ABC.

Figure 8.4: The four cases of Algorithm 8.2 for mining margin-closed Chords. The support of the Chords AB and AC is compared to the support of the common super-Chord ABC.

similar to the Algorithm 8.1 described in Chapter 8.3.5. The main difference is that no intervals within the gap are deleted, as Chords are allowed to overlap.

8.5 Finding Phrases

Task 8.5. Mining Phrases

Description	The tasks of finding (margin-closed) Phrases in a symbolic interval sequence is the creation of k (margin-closed) Phrases and the symbolic interval sequence of the occurrences thereof.
Input	Symbolic interval sequence $\mathcal{C}$ representing occurrences of a set of Chords.
Parameters	Minimum support of a Phrase. Minimum length of the paths in a Phrases. Flag whether to mine closed Phrases. Threshold α for mining margin-closed Phrases.
Output	k Phrases P_i and a symbolic interval sequence $\mathcal{P}$.

Phrases represent the temporal concept of partial order. There are many algorithms that can mine partial order from symbolic time series and sequences in the form of Episodes (see Chapter 6.1). They are not immediately applicable to the problem of mining partial order from interval sequences, however. We cannot transform the occurrence intervals of Chords to a symbolic time series, as was possible for the Events in the UTG (see Mörchen and Ultsch (2004)), because Chords can and often do overlap. The solution is to transform Chord occurrences to another well known data model, a sequence of itemsets.

Based on a set of itemset sequences, partial order can be mined with the method recently described in (Casas-Garriga, 2005). The original method was already described in Section 6.1. We will discuss the relevant parts in this Section, but concentrate on the additions and modifications needed to mine Phrases instead of partial orders of itemsets. The interested reader is referred to (Casas-Garriga, 2005) for the theoretic background and algorithmic details.

Algorithm 8.3 shows the high level overview of our proposal to find closed Phrases from the interval sequence of Chords, the details of each step will be described in the forthcoming sections. First the data model conversion to an itemsets interval series is performed (Line 1, Chapter 8.5.1). Next, the single long itemset interval series needs to be implicitly converted to a set of shorter sub series by creating a sequence of intervals (Line 2). This is necessary, because the algorithms for mining closed sequential patterns and closed partial order require a set of itemset sequences as their input. Each itemset sequence is called a transaction and support is defined as the number of transactions in which a pattern occurs. For now we assume such a segmentation is given and discuss possible ways to obtain one in Chapter 8.5.5.

For the mining of closed sequential patterns (Line 3, Chapter 8.5.2) any standard algorithm, e.g., CloSpan (Yan et al., 2003), can be used with only slight modifications. The mining of margin-closed partial orders (Line 4, Chapter 8.5.3) is the largest deviation from the methods of (Casas-Garriga, 2005). We show that the problem of mining partial orders can again be viewed as an itemset problem similar to the mining of Chords. We thus use another modified version of the CHARM algorithm. For the last step of converting a set of sequences into a partial order (Chapter 8.5.4) necessary conditions are given in (Casas-Garriga, 2005). We describe a straightforward construction for the final Phrase representation using these conditions.

Algorithm 8.3 High level algorithm to find margin-closed Phrases.

Input:

Symbolic interval sequence $\mathcal{C}$.
Minimum duration of Chord fragments in Phrase δ.
Minimum Phrase support sup_{min}, default $sup_{min} = 1$.
Minimum path length in Phrase s_{min}, default $s_{min} = 2$.
Threshold α determining margin-closedness, default $\alpha = 0.1$.

Output:

Set of Phrases and symbolic interval sequence $\mathcal{P}$.

1: Convert $\mathcal{C}$ to itemset interval series $\mathcal{I}$ using δ.
2: Create transaction windows $W = \{[s_i, e_i] | i = 1, ..., w\}$.
3: Mine pairs (s, T) composed of a closed sequential pattern s occuring within the transaction windows $T \subseteq W$ using sup_{min} and s_{min}
4: Create margin-closed maximal pairs (S, T) where S is a set of all closed sequential patterns occuring in all transaction windows $T \subseteq W$ using sup_{min} and α and create occurrence intervals $\mathcal{P}$ accordingly.
5: Build partial order of Chords for each set S.

8.5.1 Data model conversion

In order to apply algorithms from itemset mining, we convert the symbolic interval sequence of Chord occurrences to an itemset sequence, or to be more precise an itemset interval series (see Chapter 7.2.1). Let $\mathcal{C} = \{[\sigma_i, s_i, e_i] | i = 1, ..., N\}$ be the occurrences of a set of Chords $C = \{c_j = (\sigma_j, \lambda_j, \phi_j) | j = 1, ..., M\}$. We convert $\mathcal{C}$ to an itemset interval series by considering each interval of minimum duration where at least one Chord is valid as an itemset time interval and including

the symbols of all currently valid Chords as items to obtain $\mathcal{I}$ as defined in Equation 8.1.

$$\begin{aligned}\mathcal{I} \;=\; & \{[S_k, s_k, e_k] | S_k \subset \Sigma, \\ & \sigma_j \in S_k \Leftrightarrow \exists [\sigma_j, s_i, e_i] \in \mathcal{C} \wedge [s_k, e_k] \subseteq [s_i, e_i], \\ & s_k \in \{s_i | i = 1, ..., N\} \cup \{e_i + 1 | i = 1, ..., N - 1\} \\ & e_k \in \{e_i | i = 1, ..., N\} \cup \{s_i - 1 | i = 2, ..., N\}\} \end{aligned} \tag{8.1}$$

Example 8.4. In Figure 8.5 we give an example for the conversion of the interval sequence of overlapping Chord occurrences to a series of non-overlapping itemset intervals. The top rows show the occurrence intervals of three Tones indicated by the uppercase letters *A*, *B*, and *C*. The maximal occurrences of all Chords with at least two Tones are shown in the middle rows. Each Chord is assigned a unique symbol in form of a lower case letter (*a*-*d*). The bottom row shows the intervals between all start and end points of all Tones. On each interval an itemset is created including the symbols of all currently active Chords.

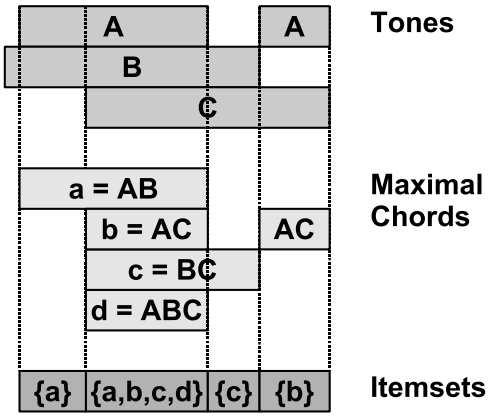


Figure 8.5: Example for the conversion of an interval sequence of Chords to an itemset interval series. Whenever a Chord interval starts or ends, a new itemset interval is created, each itemset contains the symbols of all Chords that overlap with this interval.

The parameter δ of Algorithm 8.3 for the minimum duration of Chord fragments in Phrases is omitted from the construction in Equation 8.1 for clarity, it should be set slightly lower than the minimum Chord duration.

8.5.2 Closed sequential pattern mining

The first step of the approach in (Casas-Garriga, 2005) consists of mining closed sequential patterns with an algorithm of choice. Sequential patterns describe a total order among the itemsets in the pattern. Closedness of sequential patterns is defined based upon support, i.e., a sequential pattern is closed if there is no super pattern that occurs in the same transactions.

Recall, that we excluded overlapping Chord intervals with an order relation from Phrase patterns (see Chapter 7.2.6). The reason was, that it would not be intuitive to mine the concept of coincidence twice. We therefore need to restrict any algorithm used to mine closed sequential patterns from the itemset interval series to picking at most one item per itemset. This will also speed up the mining of closed sequential patterns, because it prunes the search space to total orders of single items.

In (Casas-Garriga, 2005) it is shown how grouping of closed sequential patterns leads to closed partial orders. It is tempting to assume that margin-closed sequential patterns will also lead to margin-closed partial orders, but a simple example shows that this is not the case. Assume we

are given the 10 sequences summarized in the first three lines of Table 8.1 and want to mine margin-closed partial orders with $\alpha = 0.2$. The sequential pattern A, B, C is margin-closed as it occurs 10 times, any longer sequence occurs much less frequent (5 or 4 times), any shorter sequence occurs almost as often (10 or 9 times) according to the threshold α. The same is true for the margin-closed sequential pattern A, D, C occuring 9 times. Grouping the margin-closed sequences according to (Casas-Garriga, 2005) creates at least the groups $\{\langle A, B, C\rangle\}$ occuring 10 times and $\{\langle A, B, C\rangle, \langle A, D, C\rangle\}$ occuring 9 times. The former is a contained in the latter, but the support differs by less than α.

Sequence	Repetitions
A, D, B, C	5
A, B, D, C	4
A, B, C	1
A, D, C, E	91

Table 8.1: Example of two sets of sequences, one considering only the top three rows and one using the complete table.

We have thus observed that when using margin-closed sequential patterns, the generated partial orders are not necessarily margin-closed, there can be cases where too many groups are reported. This cannot be fixed by subsequent filtering of the results, as another example shows that is is also possible to miss margin-closed partial orders when using margin-closed sequences for grouping. Consider the extended set of 101 sequences summarized with all four rows of Table 8.1. The additional 91 occurrences of A, D, C raise the count of this sequential pattern to 100. It is not margin-closed, however, because the longer pattern A, D, C, E is almost as frequent. The exclusion of A, D, C from the grouping of sequential patterns also excludes the group $\{\langle A, B, C\rangle, \langle A, D, C\rangle\}$. But this group corresponds to a margin-closed partial order, because it is not possible to add another sequence to this group occuring in any of the transactions where it occurs, let alone in almost all. We thus have to resort to mine all strictly closed sequential patterns and incorporate the concept of margin-closedness into the grouping algorithm below.

In principle, the modification of picking only one item per itemset can be integrated in any algorithm for closed sequential pattern mining, e.g., CloSpan (Yan et al., 2003) or BIDE (Wang and Han, 2004). For simplicity of presentation we chose to modify the PrefixSpan algorithm for finding frequent sequential patterns. This algorithm does not mine closed patterns per se, we therefore included a straightforward brute force check for closedness upon adding new patterns to the result. For more efficient solutions see (Yan et al., 2003; Wang and Han, 2004).

The algorithm ClosedSequenceMining listed in Algorithm 8.4 searches for closed sequential patterns with single item itemsets. The search tree defined by the partial order of the subsequence relation is explored in a depth-first manner. The function EXTEND is recursively called given a prefix sequence s, the itemset interval series $\mathcal{I}$ and a sequence of transaction windows W for possible extensions of s. The recursion is started with an empty sequence and the initial windows in Line 1. The possible extensions of the prefix sequence by an additional item are explored in the loop starting in Line 6. All windows are checked for an occurrence of an item i. If the item is found, the remaining part of the window is stored as a new window for possible further extensions of $s \diamond i$ (Line 11). This correspond to the pseudo-projection of transaction databases in (Pei et al., 2004). Since we only include itemsets in the new window, that occur after item i, we are restricting the search to extensions of the sequential pattern by a new itemset of a single item, so called S-Step extensions (Ayres et al., 2002). The I-Step extensions, adding an item to an existing itemset of the prefix sequence are never pursued. If

the item is frequent, i.e., it occurs in more than sup_{min} of the windows, the function EXTEND is called recursively with $s \diamond i$ and the new windows in Line 15.

On each call, the EXTEND function uses the ADD-CLOSED function to check whether the current prefix s should be added to the result set of closed patterns. First, all sequences that are subsequences of s and have the same support are removed from S (Line 18). Next we search for super-sequences of s in S with the same support. If none is found s is closed and added to the result.

Algorithm 8.4 ClosedSequenceMining finds closed sequential patterns with one item per itemset.

Input:
Itemset intervals series $\mathcal{I} = \{[S_i, s_i, e_i]\}$.
Transaction windows $W = \{[s_j, e_j]\}$.
Minimum support sup_{min}.

Output:
Set S of closed sequential patterns.

1: $S :=$ EXTEND$(\mathcal{I}, W, \emptyset, \emptyset)$

EXTEND$(\mathcal{I}, W, s, S)$
2: **if** $s \neq \emptyset$ **then**
3: $S :=$ ADD-CLOSED(s, S)
4: **end if**
5: Let I be the set of all unique items in $\mathcal{I}$.
6: **for all** $i \in I$ **do**
7: $W' := \emptyset$
8: **for all** $w = [s_j, e_j] \in W$ **do**
9: **if** i appears in $\mathcal{I}|_w$ **then**
10: Let s be the start of the interval succeeding the first occurrence of i in w.
11: $W' := W' \cup \{[s, e_j]\}$
12: **end if**
13: **end for**
14: **if** $\frac{|W'|}{|W|} \geq sup_{min}$ **then**
15: $S :=$ EXTEND$(\mathcal{I}, W', s \diamond i, S)$
16: **end if**
17: **end for**

ADD-CLOSED(s, S)
18: **for all** $s' \in S$ with $sup(s) = sup(s')$ and $s' \subseteq s$ **do**
19: $S := S \setminus \{s'\}$
20: **end for**
21: **if** $\neg\exists s' \in S$ with $sup(s') = sup(s)$ and $s \subseteq s'$ **then**
22: $S := S \cup \{s\}$
23: **end if**

The incorporation of the other parameters like the maximum gap between consecutive items or the minimum size of a sequence is rather straightforward and omitted for clarity. All parameters are application dependent, it is hard to give general suggestions. The minimum support and the minimum path length influence the number of closed sequences found, lower values should be tried first. For small data sets, the minimum support could be specified as an absolute number instead of a percentage, with the according changes to the algorithm.

8.5.3 Creating margin-closed groups of sequential patterns

The mining of partial orders in (Casas-Garriga, 2005) mainly consist of creating pairs consisting of a set of closed sequential patterns and the list of transactions in which they all occur. The pairs are required to be maximal in the sense, that there is no additional sequential pattern, that occurs in all transactions and no additional transaction in which all patterns occur. Further, the sets of closed sequential patterns are required to be non-redundant, i.e., no sequential pattern is allowed to be a sub-pattern of a pattern in the same set. Algorithm 1 in (Casas-Garriga, 2005) groups the sequential patterns by first considering all patterns with the same list of transactions and then adding any pattern with a longer transaction list that does not make the group redundant. The groups are closed w.r.t. frequency and form a concept lattice.

When adding the constraints for margin-closed pairs this algorithm cannot be used anymore. For each initial group not only sequential patterns with exactly the same transaction lists need to be considered, but also patterns with only slightly shorter lists. For a valid group of patterns, the intersection of all transaction lists is allowed to be at most α percent shorter than the longest. Since all sequential patterns in a group influence the intersection, the search space for the groups consists of all 2^k combinations of k closed sequences. This corresponds to the problem of finding margin-closed itemsets, if we consider each closed sequential pattern an item and each group of sequential patterns an itemset. We can again adapt the CHARM (Zaki and Hsiao, 2005) algorithm to tackle this problem efficiently.

The algorithm ClosedPhraseMining listed in Algorithm 8.5 searches for margin-closed groups of closed sequential patterns. Each group will then be merged into a Phrase representing the partial order of a set of Chords in the next step. Only minor modifications needed to be made to the CHARM algorithm. Each group of closed sequential patterns is required to be non-redundant. Instead of checking this upon each combination of a group with possible extensions, we decided to automatically include all subsequences of a sequence in a group and remove this redundancy in a post-processing step. Further, the concept of margin-closedness was incorporated.

The search tree defined by the partial order on Phrases is explored in a depth-first manner. The function EXTEND is recursively called given a prefix Phrase p representing a set of sequential patterns and a set of super-Phrases. The recursion is started with an empty Phrase and all Phrases obtained by using each sequential pattern as a seed and adding all its subsequences in the loop starting in Line 2. The prefix group p is extended by all combinations of super-Phrases from the set P in the double loop starting in Line 16. The variable $\hat{p}_i$ (Line 17) stores the super-set of p initially only extended by p_i. The extensions are filtered by the minimum support in Line 20. The core part of the algorithm is identical to Algorithm 8.2 checking for margin-closedness and pruning the search where possible. In Line 35 the current Phrase $\hat{p}_i$ is added to the set of closed Phrases if it is not subsumed by an already found closed Phrase. The function EXTEND is called recursively in Line 37. The recursion stops as soon as the set P is empty, each Phrase is made non-redundant in the post-processing starting in Line 11.

The mining of margin-closed Phrases is the computationally most demanding task of time series knowledge mining. Compared to the number of Tones determining the complexity of the CHARM variant in Algorithm 8.2, for Algorithm 8.5 this corresponds to the number of closed sequential patterns, which will usually be much larger. Care should be taken to keep the number of closed sequential patterns small by imposing constraints on minimum length and minimum frequency. As already explained in Chapter 8.5.2 we cannot resort to using margin-closed sequential patterns. But using higher values of α for the margin-closedness of Phrases will dramatically reduce the search space and the runtime of Algorithm 8.5.

Algorithm 8.5 ClosedPhraseMining finds margin-closed Phrases by depth-first search.

Input:

Set S of margin-closed sequential patterns s with transaction lists $t(s)$.
Minimum Phrase support sup_{min}, default $sup_{min} = 1$.
Threshold α determining margin-closedness, default $\alpha = 0.1$.

Output:

Set of Phrases and symbolic interval sequence $\mathcal{P}$.

$P := \emptyset$
for all $s \in S$ **do**
  $p := \{s\}$
  for all $s' \in S$ **do**
    if $s' \subset s$ **then**
      $p := p \cup \{s'\}$
    end if
  end for
end for
$R :=$ EXTEND$(\emptyset, P, \emptyset)$
for all $p \in R$, $s \in p$ **do**
  if $\exists s' \in p \; s \subset s'$ **then**
    $p := p \setminus \{s\}$
  end if
end for

EXTEND(p, P, R)
for all $p_i \in P$ **do**
  $\hat{p}_i := p \cup p_i$
  $\hat{P} := \emptyset$
  for all $p_j \in P$ with $j > i$ **do**
    if $sup_\delta(\hat{p}_i \cup p_j) \geq sup_{min}$ **then**
      if $\frac{sup_\delta(\hat{p}_i \cup p_j)}{max(sup_\delta(\hat{p}_i), sup_\delta(p_j))} \geq 1 - \alpha$ **then**
        $\hat{p}_i := \hat{p}_i \cup p_j$
        $P := P \setminus \{p_j\}$
      else if $\frac{sup_\delta(\hat{p}_i \cup p_j)}{sup_\delta(\hat{p}_i)} \geq 1 - \alpha$ and $\frac{sup_\delta(\hat{p}_i \cup p_j)}{sup_\delta(p_j)} < 1 - \alpha$ **then**
        $\hat{p}_i := \hat{p}_i \cup p_j$
      else if $\frac{sup_\delta(\hat{p}_i \cup p_j)}{sup_\delta(\hat{p}_i)} < 1 - \alpha$ and $\frac{sup_\delta(\hat{p}_i \cup p_j)}{sup_\delta(p_j)} \geq 1 - \alpha$ **then**
        $\hat{P} := \hat{P} \cup \{\hat{p}_i \cup p_j\}$
        $P := P \setminus \{p_j\}$
      else
        $\hat{P} := \hat{P} \cup \{\hat{p}_i \cup p_j\}$
      end if
    end if
  end for
  if $\forall p \in R$ with $\hat{p}_i \subseteq p$ holds $\frac{sup_\delta(p)}{sup_\delta(\hat{p}_i)} < 1 - \alpha$ **then**
    $R := R \cup \{\hat{p}_i\}$
  end if
  $R :=$ EXTEND$(\hat{p}_i, \hat{P}, R)$
end for

8.5.4 Creating partial order from group of sequential patterns

The final step for mining Phrases is the conversion of a margin-closed set of sequential patterns into a partial order structure. Algorithm 2 in (Casas-Garriga, 2005) describes how one can determine for a pair of positions from two sequences, whether they should be joint in the partial order structure. The idea is to exactly preserve the paths defined by the sequential patterns. Each sequential pattern must be present in the partial order as a path and no additional maximal paths are allowed.

The algorithm MergeSequences in Algorithm 8.6 completes the task of creating the partial order. C is a vector of symbols that will hold all unique nodes of the partial order. E is a list of pairs that will hold all edges of the partial order. Both are initialized with the empty set in Lines 1-2. For all positions of all sequences pointers are initialized in Line 5 that will later point to the nodes C of the partial order. The outer double loop of the algorithm processes all free positions of all sequences in Lines 8-9. The current symbols is added to the end of C (Line 10) while the function ADD takes care of updating the pointer (Line 21) and adding edges to the predecessor and successor (Lines 23-26) in the sequence if they are already present in C. The inner double loop compares all positions of all other sequences with the position of the current sequence s for path preservation. The function ADD is called again to set the pointer to the same element of C and add edges accordingly.

The complexity of this algorithm is rather high. It is quadratic in the number of sequences and in the average length of the sequences. In practice both are rather small, so this is not a big issue.

8.5.5 Windowing

The long itemset series $\mathcal{I}$ needs to be segmented into several short series to enable the concept of transactions needed for mining closed sequential patterns and closed Phrases. The windows can be overlapping or non-overlapping.

Ideally, non-overlapping windows are given a priori or suggested by the data. If there are very regular occurrences of a Chord, windows could be placed between successive occurrences. For the Skating data (see Chapter A.1) the separate movement cycles are easily recognized this way. For the Video data (see Chapter A.2) there was already a set of interval series corresponding to different scenarios. In other cases large gaps between successive Chords could be used to place window boundaries irregularly. The segmentation method for symbolic sequences described in (Cohen and Adams, 2001) could possibly be used to find meaningful windows of the Chord sequence.

Non-overlapping windows have the advantage of producing less and non-redundant patterns. If the placement of the window boundaries is not justified, however, pattern instances are missed if they stretch over two or more windows. With overlapping windows of a fixed size shorter sequential patterns will have higher support, because they are present in more windows. This is a well known problem, e.g., in Episode mining (see Chapter 6.1). Possible solutions include using minimal occurrences (Mannila and Toivonen, 1996), unbounded windows (Casas-Garriga, 2003), or allowing only patterns that start with an with an item from the first itemset in the window (Chen et al., 2005). In all cases the concept of transactions and possible co-occurrences within a transaction is lost, however, and the method of Chapter 8.5.3 cannot be applied using the original sliding windows. Instead, new transaction windows need to be generated. Using each maximal occurrence as one transaction window and checking which patterns are also contained within larger windows leads to an assignment of closed sequences to transactions that can be used to perform the grouping.

Algorithm 8.6 MergeSequences creates a Phrase representing a partial order the paths of which correspond to the input set of sequences.

Input:
Set of sequences S.

Output:
Chords C and edges E of Phrase.

```
1: C := ∅
2: E := ∅
3: for all s ∈ S do
4:     for i = 1 to |s| do
5:         p_s(i) := ∅
6:     end for
7: end for
8: for all s ∈ S do
9:     for ∀i = 1, ..., |s| with p_s(i) = ∅ do
10:        C(|C| + 1) := s(i)
11:        ADD(p_s, i, |C|)
12:        for all s' ≠ s ∈ S do
13:            for ∀j = 1, ..., |s'| with p'_s(j) = ∅ do
14:                if s(p(i)) and s'(q(j)) are path preserving then
15:                    ADD(q_s', j, |C|)
16:                end if
17:            end for
18:        end for
19:    end for
20: end for

    ADD(p_s, i, c)
21: p_s(i) := c
22: if i > 0 ∧ p_s(i − 1) ≠ ∅ then
23:     E := E ∪ (C(p_s(i − 1)), C(c))
24: end if
25: if i + 1 ≤ |s| ∧ p_s(i + 1) ≠ ∅ then
26:     E := E ∪ (C(c), C(p_s(i + 1)))
27: end if
```

Chapter 9

Evaluation

In this chapter we demonstrate the power of the new framework for mining temporal knowledge in multivariate time series and time interval sequences. We first describe two data sets in Chapter 9.1 and present the result of the TSKM in Chapter 9.2. We describe how the various parameters were selected during the mining process. The resulting patterns represent meaningful descriptions of temporal phenomena in the data. For the larger data set we analyze the influence of important parameters on the number of patterns returned and the effort of the mining algorithms.

In the subsequent three sections we describe results obtained using Allen's relations (Chapter 9.3), the UTG (Chapter 9.4), and Hidden Markov Models (Chapter 9.5) on the same data sets when applicable. In Chapter 9.6 we compare the TSKM with the other methods based on the experimental findings.

9.1 Datasets

9.1.1 Skating data

The Skating data was collected from tests in professional In-Line Speed Skating by Dr. Olaf Hoos[1]. The goal of the data analysis was to identify patterns explaining the correspondence of limb movement and muscle activation. The understanding of complex muscle coordination is an important goal in biomechanics and human movement science. The knowledge gained from the muscle activation patterns can be used to further analyze the underlying mechanisms of an athletic performance and help to optimize the training process. In other areas, as rehabilitation medicine, explanations about limb movement and muscle activation patterns can be used to detect and describe anomalies and help in the development of treatments. In robotics the abstraction to interpretable rules can help to develop more realistic movement models. The coordination process can be studied by observing complex, often cyclic movements, which are dynamically repeated in an almost identical manner.

An athlete performed a standardized indoor test at a speed of 7,89m/s on a large motor driven treadmill. EMG and kinematic data of the right body side were measured for 30 seconds, corresponding to 19 complete movement cycles. The obtained multivariate time series was converted to a 5-dimensional symbolic interval series. In Chapter A.1 the preprocessing of the numerical values and the creation of symbolic time interval series is described.

Three interval series describe the activation based on EMG measurements of the muscles mainly responsible for forward propulsion: Medial Gastrocnemius (*Gastroc*), Vastus Medialis (*Vastus*), and Gluteus Maximus (*Gluteus*). The activation states were labeled *low*, *high*, and

[1]Department of Sports Medicine, Philipps-University Marburg, 35032 Marburg, Germany

very high. One series describes the foot contact with states *on* and *off* obtained from a pressure sensor in the skate, called *Trigger*. The last series describes the movement phase obtained by clustering a 6-dimensional time series with angular displacements and angular speeds of the hip, knee, and ankle joints. The phases were labeled *swing*, *glide+push*, and *recovery*.

9.1.2 Video data

The Video data was first used and described in the LEONARD system for recognition of visual events from video camera input (Siskind, 2001). It is available as an online appendix of (Fern et al., 2002)[2]. Different scenarios involving a human hand moving colored blocks were first recorded on video. Using image segmentation and motion tracking algorithms a description of each video in terms of primitive events valid on time intervals was generated (Siskind, 2001). Automated analysis of video data is useful for applications like surveillance or quality assurance.

We preprocessed the descriptions by normalizing the argument order, filtering out some redundant descriptions, and merging scenarios that were equivalent, see Chapter A.2 for details. The resulting Tones are listed in Table 9.1. The eight qualitatively different scenarios are *pick-up*, *put-down*, *stack*, *unstack*, *assemble*, and *disassemble* with 30 repetitions each, and *move-left* and *move-right* with 15 repetitions each.

Label	Description
contacts-blue-green	The blue block and the green block have contact.
contacts-blue-red	The blue block and the red block have contact.
contacts-green-red	The green block and the red block have contact.
attached-blue-hand	The hand holds the blue block.
attached-hand-red	The hand holds the red block.
attached-blue-green	The blue block and the green block have contact and are part of a stack where the top block is held by the hand.
attached-blue-red	The blue block and the red block have contact and are part of a stack where the top block is held by the hand.
attached-green-red	The green block and the red block have contact and are part of a stack where the top block is held by the hand.

Table 9.1: Description of all Tones from the Video data.

In (Fern et al., 2002) each scene is described with a logical formula found by inductive logic programming in a supervised process. We use the data in an unsupervised manner and use the ground truth on the scenarios only for evaluation purposes. This is an example of interval data that was not obtained from times series. Since no additional information on the eight Tones was available, we can view this as eight different Aspects each with a single state that is valid during the Tone intervals and not valid otherwise.

9.2 Efficacy and Efficiency of TSKM

9.2.1 Skating data

We first mined Chords with a minimum duration of 50ms. Shorter phenomena related to muscle activation are physiologically not plausible. The minimum size was set to 3, the maximum size is naturally bounded by the five dimensions of the interval series. With a minimum support of 2% and a minimum count of eight a total of 70 Chords was found. Restricting the patterns to closed Chords reduced the number to 60, mining margin-closed Chords with $\alpha = 0.1$ returned 18

[2] `ftp://ftp.ecn.purdue.edu/qobi/ama.tar.Z`

patterns. At this point it is difficult to decide automatically whether rare large or small frequent Chords are better. The lattice of the Chords annotated with frequency (#) and support (%) was presented to the expert, who selected and labeled nine Chords as the most interesting. The selected Chords are bold shown in Figure 9.1.

Using these nine Chords and non-overlapping windows corresponding to the 19 movement cycles of the experiment we found 44 closed sequential patterns with a minimum length of three and a minimum frequency of eight. Merging the closed sequences into groups according to Casas-Garriga (2005) resulted in 25 closed Phrases, the Algorithm 8.5 from Chapter 8.5.3 with $\alpha = 0.1$ returned 15 margin-closed Phrases. Again we presented the lattice of Phrases, parts of which are shown in Figure 9.2, to the expert. Two paths of sub-Phrase relationship are visible between the most general pattern at the top and the most specific pattern at the bottom. From top to bottom, Chords are successively added to the partial order patterns making them more specific but less frequent.

The expert selected the top and bottom Phrases as the most interesting. The most general Phrase at the top shows the partial order of six Chords present in 18 of the 19 movement cycles. It summarizes the simultaneous occurrence of the three closed sequences of length four that are obtained by traversing all maximal paths of the Phrase. The most detailed Phrase at the bottom is present in eight cycles and represents the total order of eight out of the nine Chords.

The Phrases were considered valid and interesting by the expert. The general Phrase describes the most dominant components of the cyclical movement, the specific Phrase offers additional information about the successive activation of the three muscles and the exact order relation of the Chords occuring before *active gliding*. Of particular importance is the connection of external variables describing the limb movement with the internal observations on muscle activation made by the Chord patterns.

The pruning by margin-closedness largely reduced the number of patterns so they could easily be analyzed manually to trade off pattern size vs. pattern frequency. We analyzed the search space in dependence of the parameters to quantify the pruning effects. In Figure 9.3(a) the number of Chords found is shown for different values of the parameter α determining margin-closedness. The value of 0.1 as used for the analysis above, effectively prunes the size of the result down to a manageable number of patterns that can be analyzed manually. Using lower values would result in a dramatic increase of patterns, using higher values results in less than five patterns for $\alpha > 0.18$. We also measured the number of recursive calls to the EXTEND function of Algorithm 8.2 as an estimate of the runtime of the algorithm independent of implementation and hardware. As is visible in Figure 9.3(b) the concept of margin-closedness does not only reduce the size of the result set, it also effectively prunes the search space during mining and speeds up the computation. This is because changin the notion of strict closedness to margin-closedness more often enables the activation of the cases 1-3 in Algorithm 8.2 that have a pruning effect. For $\alpha = 0.2$ less than 10% of the functions call of strict closed Chord mining are needed.

Similar observations were made for Phrase mining with Algorithm 8.5. The 25 strictly closed Phrases already offer a good summarization of the 44 closed sequences, but using $\alpha = 0.1$ further reduces this down to 15, using $\alpha = 0.2$ only six dominant Phrases remain. Again the pruning of the result set is closely connected to the number of recursive function calls in the algorithm, because both Chord and Phrase mining are performed by modified version of the CHARM algorithm. For $\alpha > 0.2$ less than half the number of function calls than for strictly closed Phrases are needed. Interestingly, the number of function calls is not strictly decreasing with larger values of α. This is likely caused by the data dependent structure of the search space. A general downward trend is clearly visible, though.

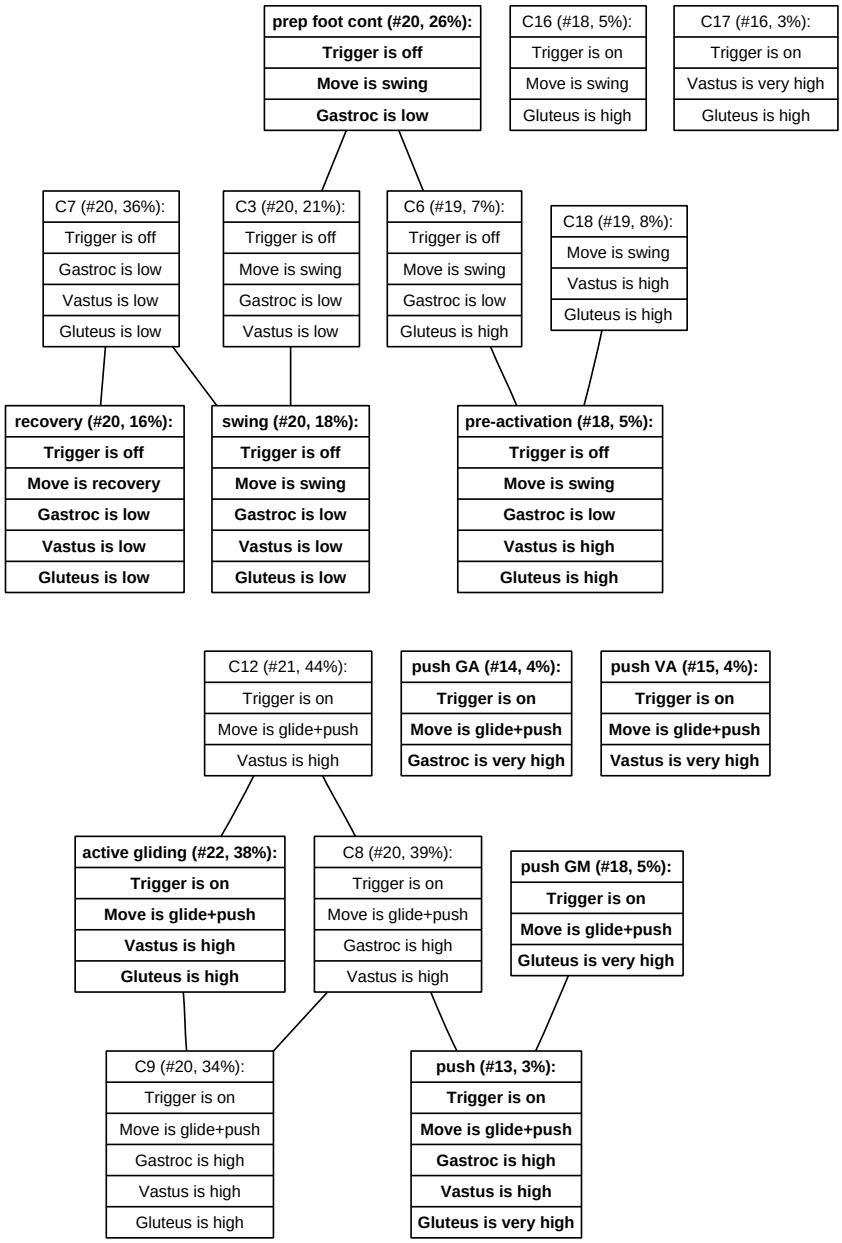


Figure 9.1: The lattice of all margin-closed Chords from the Skating data. The expert labeled and selected the nine Chords shown bold. The other Chords were either too similar to one of the selected Chords or did not represent an interesting relation of Tones.

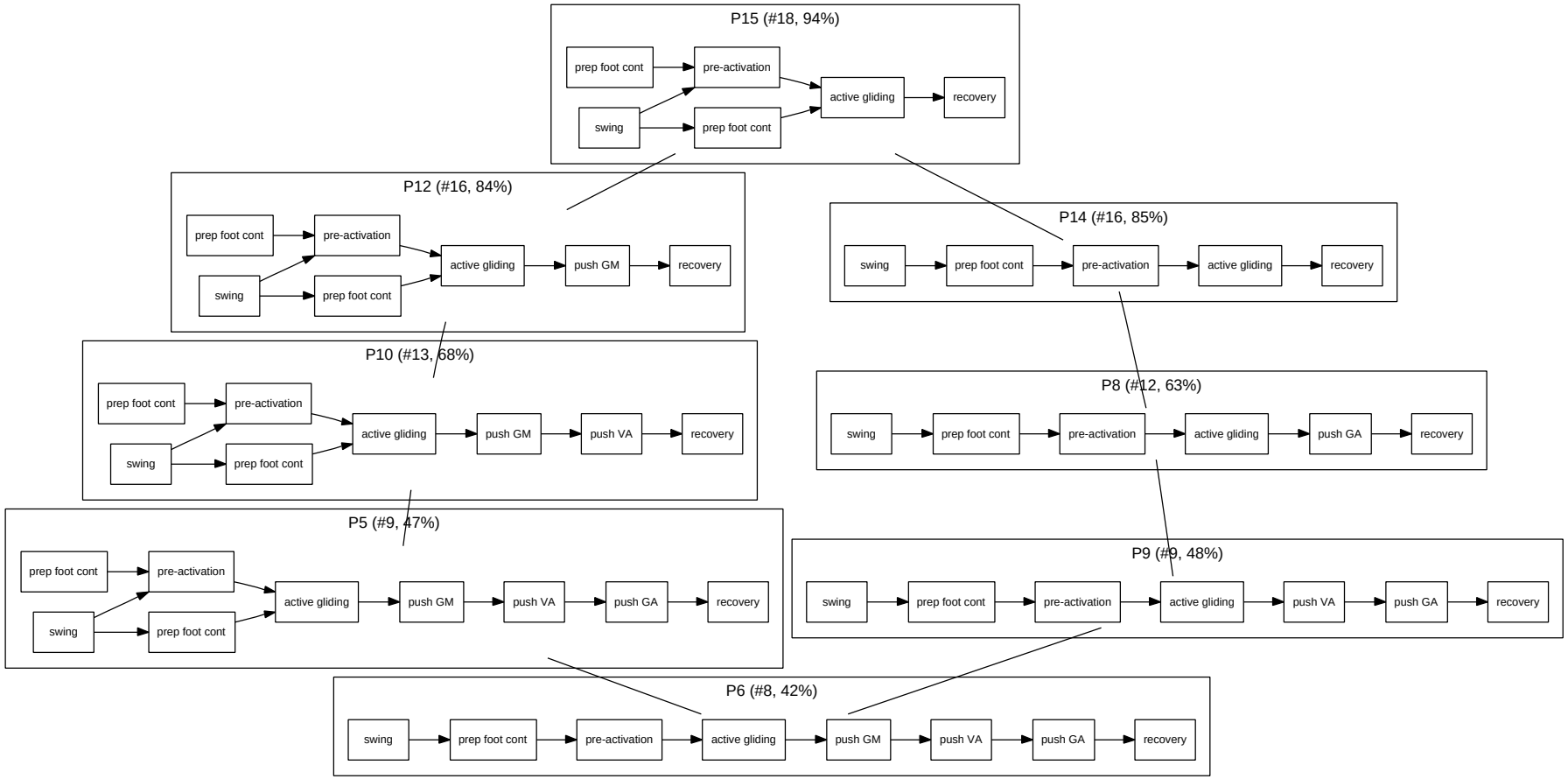
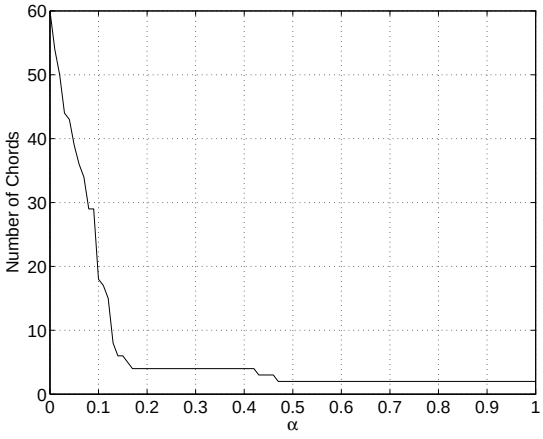
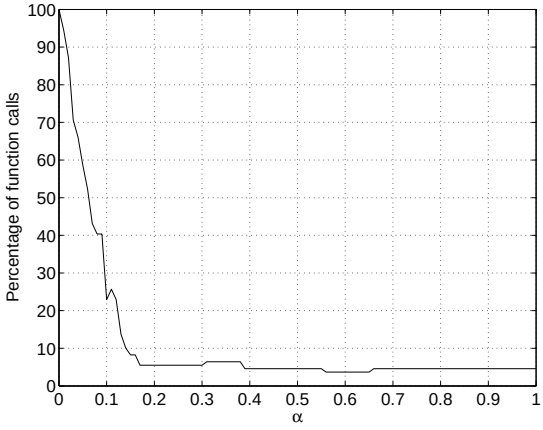


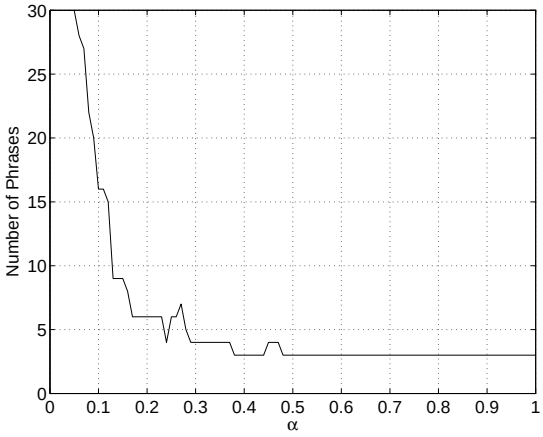
Figure 9.2: The lattice of some margin-closed Phrases for Skating data. The topmost and bottommost Phrases were selected by the expert as most interesting. They provide a general and a detailed descriptions of the movement cycle during Inline-Speed Skating, respectively.



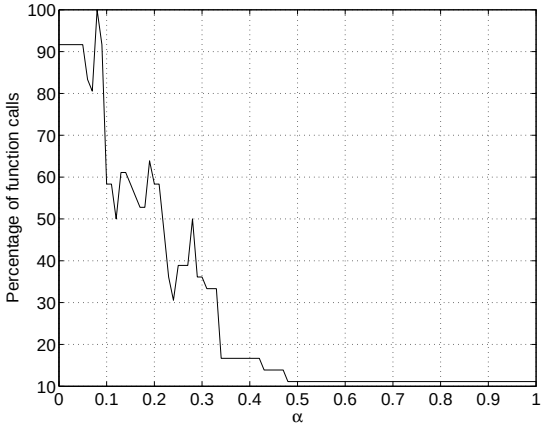
(a) Number of Chords found



(b) Percentage of recursive function calls in Chord mining.



(c) Number of Phrases found



(d) Percentage of recursive function calls in Phrase mining.

Figure 9.3: Effects of different values of α on the size of the result and the complexity of the search process for Chord and Phrase mining. Larger values of α effectively reduce the amount of patterns reported and improve efficiency of the mining algorithms at the same time.

9.2.2 Video data

For the Video data no prior knowledge for the selection of the minimum duration of Chords was available. Figure 9.4 shows the distribution of Tone durations in the Video data. Since the majority of Tones has a duration of 10 or more ticks we chose five as the minimum duration for Chords. The minimum size was set to one, allowing single Tones as trivial Chords. A single Tone like the hand grabbing a block already represents an important fact that does not necessarily need to be combined with other currently valid Tones. We further chose an arbitrary but small minimum support of 1% and a minimum count of 10, considering that the *move-left* and the *move-right* scenarios have only 15 repetitions in the data.

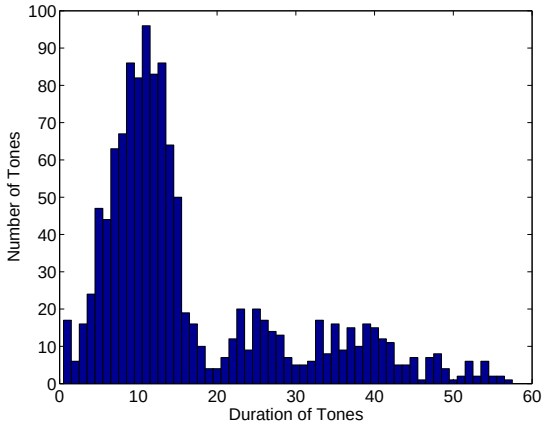


Figure 9.4: Histogram showing the distribution of Tone durations in the Video data.

According to these parameters, there are 19 Chords in the data, all of which are closed. A value of $\alpha = 0.1$ for margin-closedness reduced the number of patterns down to 15. We further dropped three trivial Chords of size one not involving the hand, but rather one of the *contacts* Tones. These Tones are only interesting when combined with a hand action. The lattice of the remaining Chords is shown in Figure 9.5. Since the Chords contain just a few Tones, we did not relabel them, but will use the full Tone listing for the representation of Chords in the Phrases below.

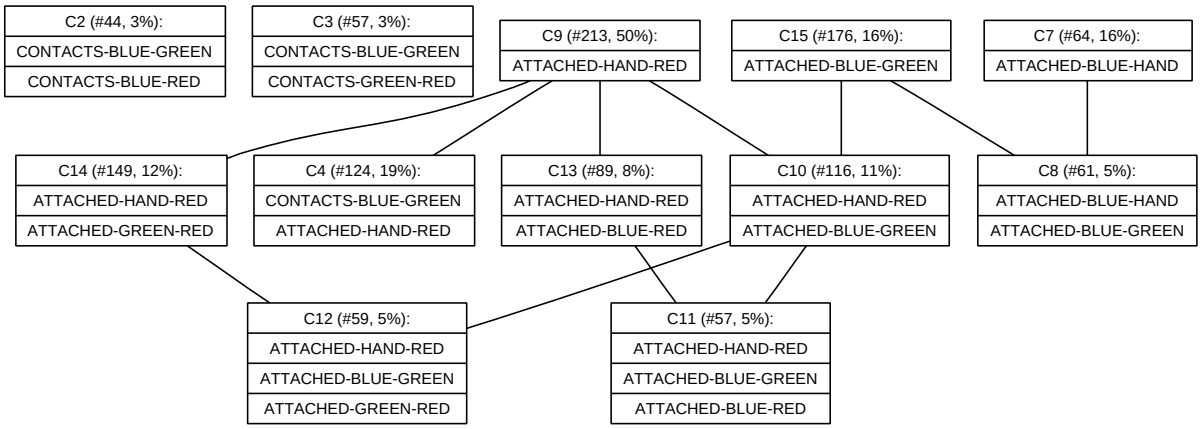


Figure 9.5: The lattice of the most interesting margin-closed Chords from the Video data. The three trivial Chords in the upper right represent the hand holding one of the three blocks. The two most specific Chords with three Tones describe a stack of three blocks with the hand holding the topmost block.

The Chords C2 and C3 in the upper left represent a stack of three blocks not touched by the hand. Without looking at the original Video data, it is unclear at this point however, which block sits on top and which at the bottom. For C2 we only know that blue is in the middle, because it is an argument of both *contacts* relations. The trivial Chord C9 describes the hand holding the red block. This Chord has very high support, covering 50% of all time points and has several super-Chords. C13 and C14 describe the hand holding the red block, while it sits on top of the green or blue block, respectively. Both have further super-Chords C11 and C12 where yet another block is part of the stack. Note, how the support of the super-Chords is always at least 10% smaller than that of any immediate sub-Chords, according to $\alpha = 0.1$.

The occurrences of the Chords within each known scenario are listed in Table 9.2. The correspondences are striking. All Chords except C9 appear only during some scenarios and all except C2 seem to appear in almost all repetitions[3]. C2 is only present in about half of the *assemble* scenarios. C15 seems to appear twice within most *assemble* and *disassemble* scenes. The Chord patterns are not sufficient, however, to discriminate the scenarios, because they always appear in at least two different scenes. There are four pairs of scenarios that are semantically reversed version of one another. In order to distinguish each pair, we need patterns expressing an order among the Chords.

Scenes	Chords											
	C2	C3	C4	C7	C8	C9	C10	C11	C12	C13	C14	C15
pick-up	0	0	0	0	0	30	0	0	0	0	30	0
put-down	0	0	0	0	0	30	0	0	0	0	31	0
move-left	0	0	0	0	0	15	0	0	0	15	15	0
move-right	0	0	0	0	0	15	0	0	0	14	14	0
stack	0	28	30	0	0	30	30	0	30	0	30	30
unstack	0	29	30	0	0	30	30	0	29	0	29	30
assemble	14	0	34	34	31	33	26	27	0	30	0	56
disassemble	30	0	30	30	30	30	30	30	0	30	0	60

Table 9.2: Occurrences of Chords in each of the scenes of the Video data.

During the conversion of the interval sequence of Chords to an itemset interval series we applied an additional filtering step for this data set. From Figure 9.5 it is clear that in this application more specific Chords are always more interesting. We thus removed all sub-Chords of a larger Chord within each itemset. The smaller Chords are still present on intervals where they appear without any concurrent super-Chord. Using a minimum frequency of 12, 20 closed sequential patterns were found and grouped into the 10 margin-closed Phrases ($\alpha = 0.1$) shown in Figure 9.6. We added the labels of the represented scenarios to the Phrase names according to the co-occurrence analysis described below.

The occurrences of the Phrases in the known scenarios in Table 9.3 shows an almost perfect correspondence of the found Phrases with the underlying temporal phenomena in the data. For most scenes the Phrase describing the actions in the video can easily be picked by using the maximum of the corresponding row. In cases where this does not suffice it helps to look at the Phrase relations in Figure 9.6. For *move-left* we picked Phrase P10. Even though P9 also occurs in all *move-left* scenes, it is a less specific description than P10 and also describes the *pick-up* scene. The *pick-up* scene is thus a sub-scene of *move-left* because blocks of the same colors were used. Similar, for *move-right* P7 is the best description because it is more specific than the super-Phrases P8 and P6, the latter describes the *put-down* scene. For *assemble* we picked the the more specific Phrase P5 instead of the slightly more frequent P4.

[3] All occurrences within a window were counted.

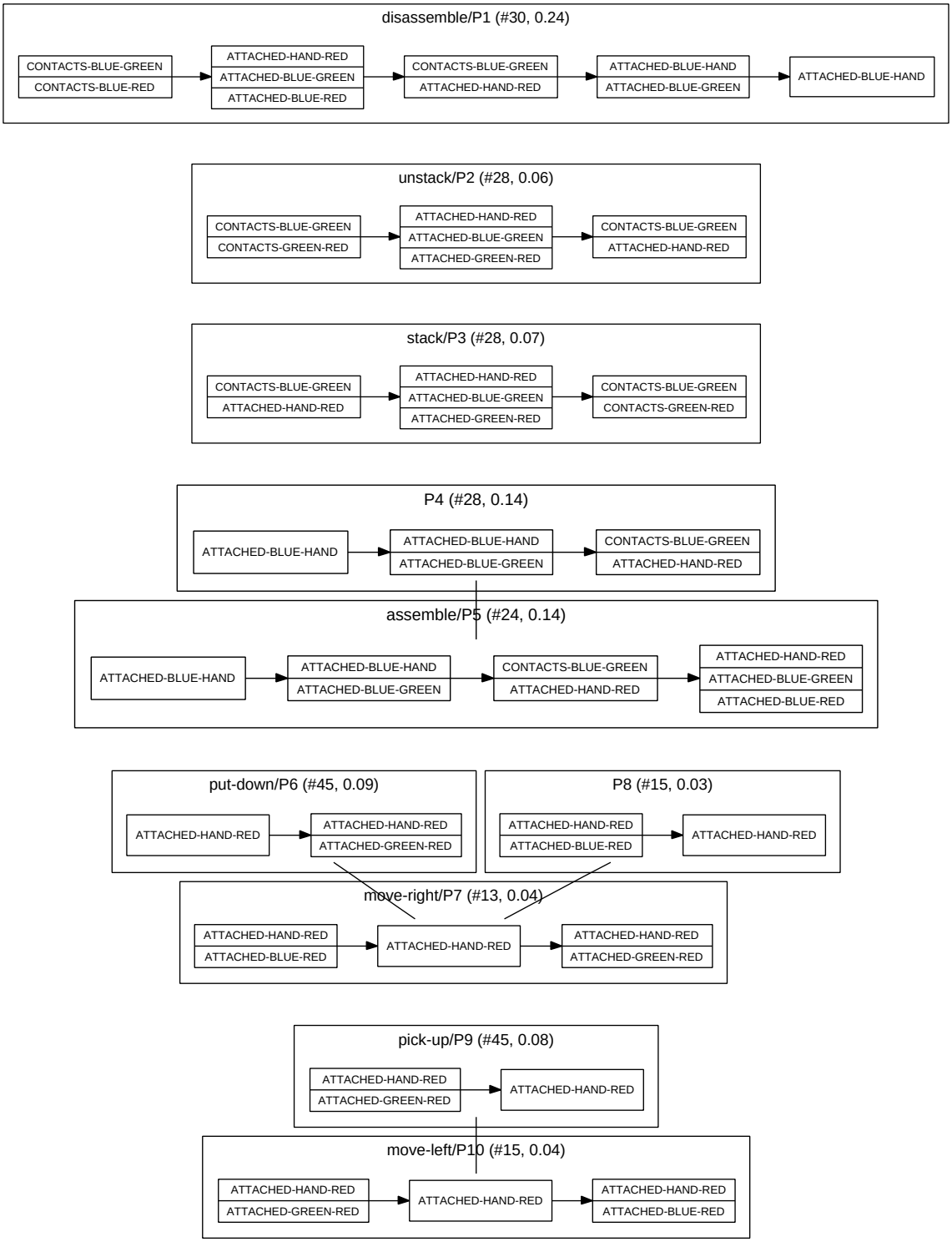


Figure 9.6: The lattice of all margin-closed Phrases from the Video data. The Phrases that best represent each scenario were labeled accordingly.

Scenes	Phrases									
	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
pick-up	0	0	0	0	0	1	0	0	30	0
put-down	0	0	0	0	0	30	0	0	0	0
move-left	0	0	0	0	0	0	0	1	15	15
move-right	0	0	0	0	0	14	13	14	0	0
stack	0	0	28	0	0	0	0	0	0	0
unstack	0	28	0	0	0	0	0	0	0	0
assemble	0	0	0	28	24	0	0	0	0	0
disassemble	30	0	0	0	0	0	0	0	0	0

Table 9.3: Occurrences of Phrases in each of the scenes of the Video data.

The selected Phrases do not only show high correspondence with the known classification of segments in the data, they further *explain* the actions in the videos. In the *pick-up* Phrase (P9) the hand first touches the red block, which is resting on the green block (*attached-hand-red* and *attached-green-red*). Next, the hand is holding the red block (*attached-hand-red*) and the relation of the red and green block has vanished. This is exactly what happens in the videos for this scene. The *put-down* Phrase (P6) represents the reversed order of the same Chords, indicating the hand holding the red block and placing it on top of the green block. The same inverse relation is observed between the two *move* scenes (P7 and P10) where the red block is placed on (taken off) the blue block. In the *stack* Phrase (P3) the hand is first holding the red block, while the green block is already resting on the blue block (*attached-hand-red* and *contacts-blue-green*). When the red block is placed on top of the stack, all three blocks and the hand are connected with three *attached* relations. The third Chord in this Phrase describes the final stack of three blocks with two *contacts* relations. The hand that has released the red block, the *attached-hand-red* has disappeared. Again *unstack* Phrase (P2) is the exact reverse of the *stack* scene. The most complex scenes, *assemble* and *disassemble* are also well described by the Phrases P5 and P1, even though for assemble the final stack of three blocks is not recognized.

9.3 Allen's Relations

In Chapter 7.3 we have shown many theoretical advantages of the TSKR over Allen's relations. We now demonstrate the problems of Allen's relations on the two data sets where mining TSKR patterns was successful. We used the pattern format of (Höppner, 2003) using all pairwise Allen's relations of the intervals within a pattern, because the compact formats are ambiguous (see Chapter 7.3). The support of patterns was measured based on counting occurrences in windows as in (Kam and Fu, 2000; Cohen, 2001) to be comparable with the Phrase mining that also relies on a given windowing. We further pruned the set of patterns applying the concept of margin-closedness in a brute force post-processing step.

9.3.1 Skating data

We searched the Skating data for patterns expressed with Allen's relations using a minimum frequency of eight and a minimum size of two resulting in 6844 patterns. Pruning by enforcing margin-closed patterns with $\alpha = 0.1$ returned 1073 patterns. The resulting pattern size distribution is shown in Figure 9.7.

There were six patterns of the largest size nine, one example of each is shown in Figure 9.8. The pattern A4 appeared in 10 out of 19 movement cycles, the others barely meet the minimum frequency of eight. The pattern visualizations were presented to the expert, because the listing

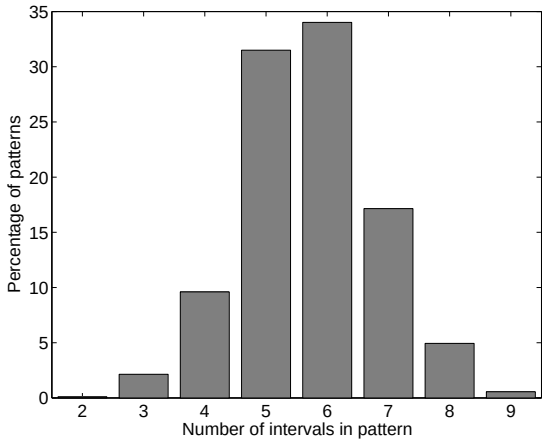


Figure 9.7: Histogram showing the distribution of sizes for margin-closed patterns expressed with Allen's relations in the Video data.

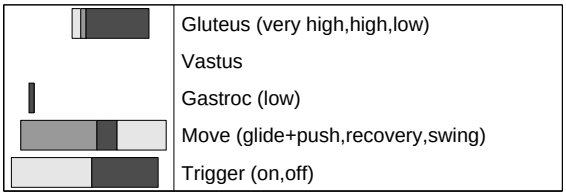
of all 36 pairwise relations did not offer a comprehensible description, see Figure 9.9 for pattern A4.

The relation of the Gluteus muscle activity to the movement phases and the trigger states, visible in many of the six patterns, was recognized as valid. But the extremely limited information provided on the states of the Vastus and Gastroc muscles, mainly responsible for forward propulsion, rendered the patterns useless to the expert. There are only two patterns containing states of the Vastus muscle. In A3 (Figure 9.8(c)) only the *very high* state is included and no information on Gastroc is given. In Figure 9.8(f) all three Vastus states are present, but no information on the foot contact (Trigger) is available. It is the connection of external information like movement phase and foot contact with internal information on muscle activity, however, that is of utmost importance for this application. The Gastroc muscle only appears with low activity, if at all. Other facts provided by the patterns, e.g., the sequence of the three movement phases in the displayed order and the two alternating Trigger states, are already obvious from the input data and offer no additional information.

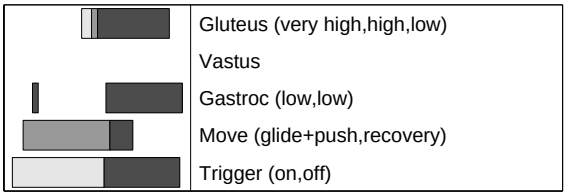
Because a correspondence of muscle activity to movement related information was sought we filtered the pattern set explicitly for patterns containing the states *high* or *very high* for the *Vastus* and *Gluteus* muscles and *on* for the Trigger. The results are listed in Table 9.4. With too strict conditions, no patterns were found at all. Only if one of the states describing the Vastus muscle to be active was not required to be part of the pattern, a few patterns with relatively low support were found. Further, the Gluteus muscle states and the foot contact were almost always related to the high or very high activation of Vastus with the *before* relation, which is not as interesting as relations with overlapping intervals in this application.

Trigger	Vastus		Gluteus		Number of	Largest
on	high	very high	high	very high	patterns	support
×		×	×	×	6	11
×	×		×	×	3	8
×	×	×		×	0	0
×	×	×	×		0	0
×	×	×	×	×	0	0

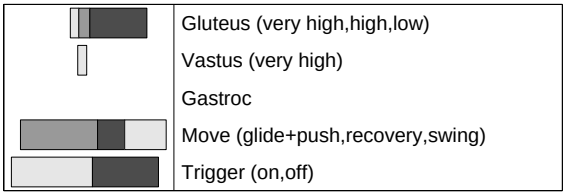
Table 9.4: Number of patterns and maximum support for Allen patterns relating active states of the two most important muscles with the foot contact.



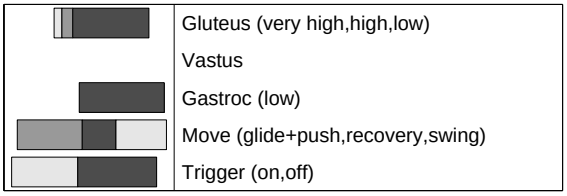
(a) A1 found 8 times



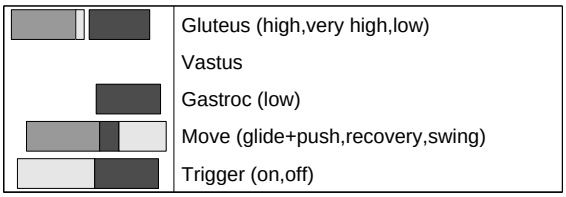
(b) A2 found 8 times



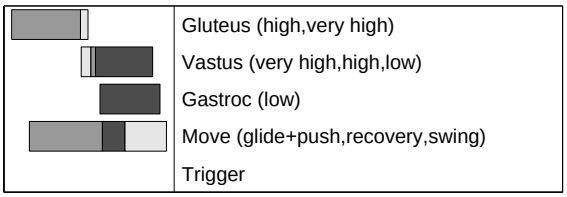
(c) A3 found 8 times



(d) A4 found 10 times



(e) A5 found 8 times



(f) A6 found 8 times

Figure 9.8: Example instances of the largest patterns found with Allen's relations in the Skating data. Each Aspect is shown in one row, the Tone labels for the intervals on the left are listed in parenthesis.

Trigger is on *overlaps* Move is glide+push
Gluteus is very high *during* Trigger is on
Gluteus is high *during* Trigger is on
Trigger is on *overlaps* Gluteus is low
Trigger is on *meets* Trigger is off
Trigger is on *before* Gastroc is low
Trigger is on *before* Move is recovery
Trigger is on *before* Move is swing
Gluteus is very high *during* Move is glide+push
Gluteus is high *during* Move is glide+push
Move is glide+push *overlaps* Gluteus is low
Move is glide+push *overlaps* Trigger is off
Move is glide+push *overlaps* Gastroc is low
Move is glide+push *meets* Move is recovery
Move is glide+push *before* Move is swing
Gluteus is very high *meets* Gluteus is high
Gluteus is very high *before* Gluteus is low
Gluteus is very high *before* Trigger is off
Gluteus is very high *before* Gastroc is low
Gluteus is very high *before* Move is recovery
Gluteus is very high *before* Move is swing
Gluteus is high *meets* Gluteus is low
Gluteus is high *before* Trigger is off
Gluteus is high *before* Gastroc is low
Gluteus is high *before* Move is recovery
Gluteus is high *before* Move is swing
Gluteus is low *overlaps* Trigger is off
Gluteus is low *overlaps* Gastroc is low
Move is recovery *during* Gluteus is low
Gluteus is low *overlaps* Move is swing
Trigger is off *overlaps* Gastroc is low
Move is recovery *during* Trigger is off
Trigger is off *overlaps* Move is swing
Move is recovery *during* Gastroc is low
Gastroc is low *overlaps* Move is swing
Move is recovery *meets* Move is swing

Figure 9.9: Listing of pairwise relations for the most frequent large Allen pattern A4 found in the Skating data.

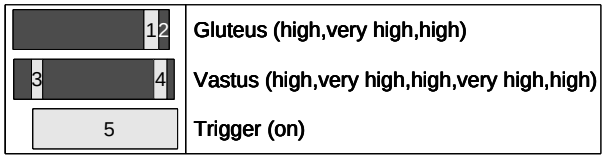


Figure 9.10: Example instance of a rare Allen pattern that contains information on Gluteus, Vastus, and Trigger. The numbered intervals are referred to in Table 9.5.

There seemed to be a mutually exclusive relationship between internal and external information that is contrary to the aim of the study. We looked at the relative positioning of the interesting Gluteus, Vastus, and Trigger intervals in each movement cycle. An example is shown in Figure 9.10. The first very high activation state of Vastus overlaps the foot contact (Trigger on) and towards the end of the foot contact, first the Gluteus muscle and then the Vastus muscle show very high activity followed by high activity. This pattern represents the expected temporal relations of the very high states and the foot contact. It only occurs two times with exactly the same relationships according to Allen, however. Many different fragmented versions are present where either one of the short high activation intervals is missing, or one or more of the overlaps relations change to the during or before relations. The occurrences considering all possible combinations of the numbered intervals from Figure 9.10 are listed in Table 9.5.

		3 missing	3 overlaps 5	3 during 5
1 or 4 missing		3	1	1
1 before 4		1	0	0
1 overlaps 4	4 overlaps 2	4	2	2
1 overlaps 4	4 during 2	2	0	2
1 during 4		0	0	1

Table 9.5: Occurrences of fragments of the Allen pattern shown in Figure 9.10 by considering similar alternatives for the relations between the intervals.

9.3.2 Video data

For the video data we searched patterns expressed with Allen's relation using a minimum size of two and a minimum frequency of 10. This resulted in 363 patterns, 174 of which were margin-closed for $\alpha = 0.1$. This large amount of patterns could not be analyzed manually, some filtering needed to be applied.

We first looked at the 14 largest patterns with five or more intervals. Patterns of these sizes were only observed within the *(dis)assemble* scenes as shown in Table 9.6. From the valid TSKR patterns for the scenes *pick-up*, *put-down*, and *move* (see Figure 9.6) we know, that there are only three qualitatively different intervals, so no Allen patterns of size five or more were to be expected. The *(un)stack* scenes however, consist of six intervals (counting the *attached-hand-red* only once, since it is present in two consecutive Chords) and should have been found if the pairwise interval relations were consistent enough for the frequency threshold.

The patterns L8 and L9 describe fragments of the *assemble* scenario. The rules for these patterns are listed in Figure 9.11 and Figure 9.12. Five out of ten conditions are equal, the other differ either in the relation or in the participating intervals.

A graphical representation of one example for each pattern is given in Figure 9.13. Going from left to right, both patterns start with the hand holding the blue block when only the *attached-blue-hand* interval is present in the second row. The placing of the blue on the green block is described with the next interval in the top rows and Allen's relation *attached-blue-hand*

Scenes	Allen patterns													
	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14
pick-up	0	0	0	0	0	0	0	0	0	0	0	0	0	0
put-down	0	0	0	0	0	0	0	0	0	0	0	0	0	0
move-left	0	0	0	0	0	0	0	0	0	0	0	0	0	0
move-right	0	0	0	0	0	0	0	0	0	0	0	0	0	0
stack	0	0	0	0	0	0	0	0	0	0	0	0	0	0
unstack	0	0	0	0	0	0	0	0	0	0	0	0	0	0
assemble	0	0	0	0	0	0	0	10	10	0	0	0	0	0
disassemble	21	21	21	21	20	21	20	0	0	17	17	17	11	11
size	5	5	5	5	5	5	5	5	5	6	6	6	7	7

Table 9.6: Occurrences of large Allen patterns in each of the scenes of the Video data.

attached-blue-green *ends* attached-blue-hand
attached-blue-hand *meets* contacts-blue-green
attached-blue-hand *before* contacts-blue-green
attached-blue-hand *before* contacts-blue-red
attached-blue-green *meets* contacts-blue-green
attached-blue-green *before* contacts-blue-green
attached-blue-green *before* contacts-blue-red
contacts-blue-green *before* contacts-blue-green
contacts-blue-green *before* contacts-blue-red
contacts-blue-green *equals* contacts-blue-red

Figure 9.11: Listing of pairwise relations for the large Allen pattern L8 found in Video data.

ends attached-blue-hand. The remaining intervals describe only fractions of the actions in the video, however, the subsequent placing of the red block on top of the stack is missing in both patterns.

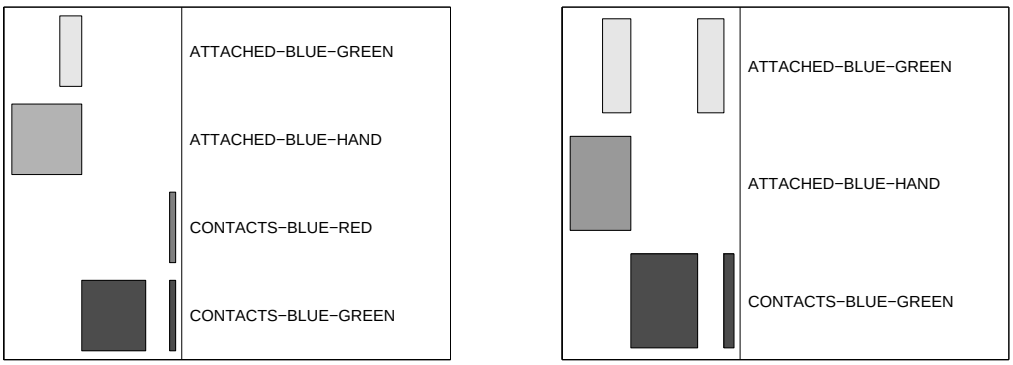
The patterns L13 and L14 shown in Figure 9.14 both almost completely describe the *disassemble* scene. L13 is started by two *contacts* relations in the two bottom rows indicating the initial stack, L14 includes only one of the two. Next, when the hand touches the stack, in both patterns three *attached* states are observed simultaneously in the top rows. When the hand lifts up the top most block, only L14 contains the concurrent *contacts-blue-green* relation of the other two blocks. Both patterns continue with the removal of the second block indicated by two concurrent *attached* actions in the second and fourth row and finish with only the hand holding one block (*attached-blue-hand*). Each pattern thus misses one interval that is contained in the other one. There is no complete pattern of size eight found, even when lowering the frequency threshold to five.

Since the largest patterns were not very useful we filtered the mining results by frequency. Given the prior knowledge on 30 scenarios for most scenes we selected all patterns with at least 24 occurrences and a minimum size of three. A total of 28 patterns were found. The confusion matrix with the known scenes is given in Table 9.7. We omitted 15 patterns that were only present in the *disassemble* scenes. They were only of size three and four and better, larger patterns have already been discovered for this scene above.

For the *pick-up* scene the patterns F8 and F9 give valid and quite similar descriptions of the actions in the scene (see Figure 9.15). First, only the *contacts-green-red* state is observed. Next, the hand is touching the stack of two blocks (*attached-hand-red* and *attached-green-red* are valid) and finally only the *attached-hand-red* state is valid. The difference between the two

attached-blue-green *ends* attached-blue-hand
attached-blue-hand *meets* contacts-blue-green
attached-blue-hand *before* attached-blue-green
attached-blue-hand *before* contacts-blue-red
attached-blue-green *meets* contacts-blue-green
attached-blue-green *before* attached-blue-green
attached-blue-green *before* contacts-blue-green
contacts-blue-green *meets* attached-blue-green
contacts-blue-green *before* contacts-blue-green
attached-blue-green *meets* contacts-blue-green

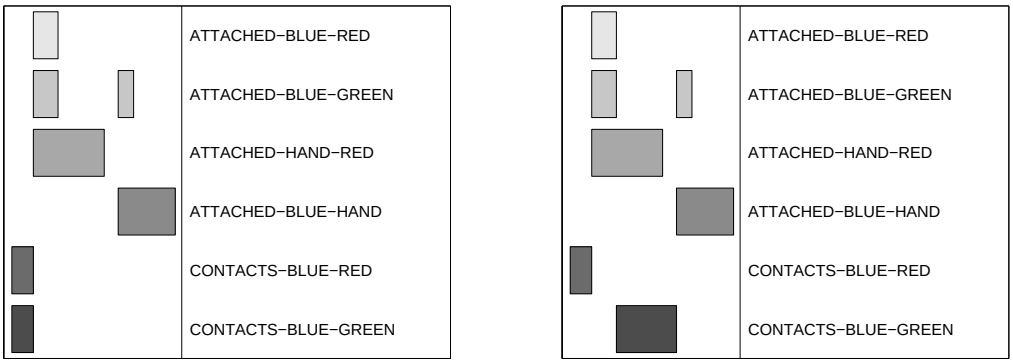
Figure 9.12: Listing of pairwise relations for the large Allen pattern L9 found in Video data.



(a) Example of L8

(b) Example of L9

Figure 9.13: Example instances the large Allen patterns L8 and L9 found in *assemble* scenes of the Video data. The graphical representations are easier to compare than the textual representations in Figure 9.11 and Figure 9.12.



(a) Example of L13 (b) Example of L14

Figure 9.14: Example instances the large Allen patterns L13 and L14 found in *disassemble* scenes of the Video data. The patterns differ only in the bottom rows of intervals describing the contact of the blue and green block. No large pattern combining L13 and L14 is found.

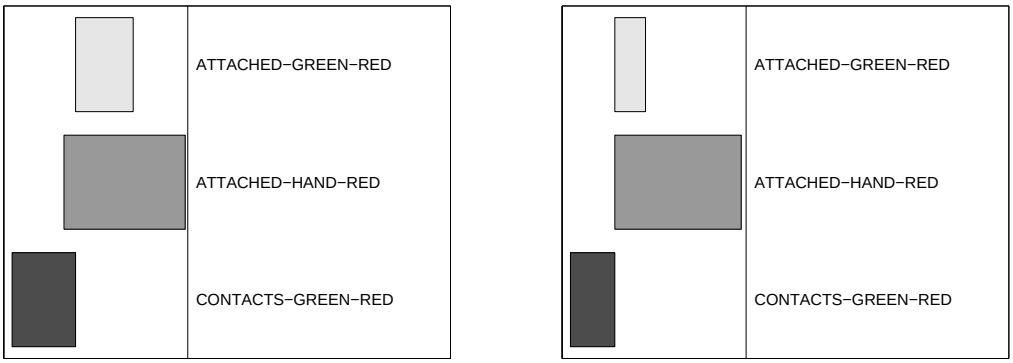
Scenes	Allen patterns												
	F2	F6	F8	F9	F10	F12	F13	F14	F15	F16	F17	F19	F20
pick-up	0	0	11	15	0	0	0	0	0	0	0	0	0
put-down	0	0	0	0	0	0	0	18	0	0	0	0	0
move-left	0	0	3	11	0	0	13	0	0	0	0	0	0
move-right	0	8	0	0	0	0	0	9	0	0	0	0	0
stack	0	0	0	0	0	20	0	16	0	0	0	0	0
unstack	27	0	12	17	0	0	0	0	0	0	0	27	0
assemble	0	0	0	0	27	5	15	0	37	36	25	0	29
disassemble	21	26	0	0	0	0	0	0	1	0	0	0	0
size	3	3	3	3	3	3	3	3	3	3	3	3	3

Table 9.7: Occurrences of frequent Allen patterns in each of the scenes of the Video data.

patterns is the exact relation of the *contacts-green-red* and *attached-hand-red* intervals. In F8 they are related by the *overlaps* relation and in F9 by the *meets* relation. This is an example of pattern fragmentation. The *meets* and *overlaps* relations are intuitively very similar in this case, because the overlapping part is only one or two time units long. Both patterns would not have been found with higher frequency thresholds, in fact they are only found because they also appear in other scenes, which makes them less distinctive. A third version where the two intervals are related with the *before* relation with a gap of only one time unit appears in the remaining four *pick-up* scenes.

Both F8 and F9 patterns also appear quite often in the *unstack* scene. This was not observed in the TSKR descriptions. The Chord describing the coincidence of *attached-hand-red* and *attached-green-red* was found in both *pick-up* and *unstack*, but not the Phrase describing the order of the former Chord and the trivial Chord *attached-hand-red*. A closer look at Figure 9.6 reveals, however, that the Phrase for *pick-up* is indeed contained in the Phrase for *unstack*. It is not a sub-Phrase by definition, however, because the 2-Chord in *pick-up/P9* is extended to a 3-Chord in *unstack/P2*.

For the *put-down* scene only one pattern (F14) shown in Figure 9.16 is observed. It is the reversed version of the patterns for the *pick-up* scene. First, only the *attached-hand-red* is



(a) Example of F8 (b) Example of F9

Figure 9.15: Example instances the frequent Allen patterns F8 and F9 found in *pick-up* scenes of the Video data. The two patterns are visually very similar, but the relations of the intervals are different due to slightly shifted interval boundaries.

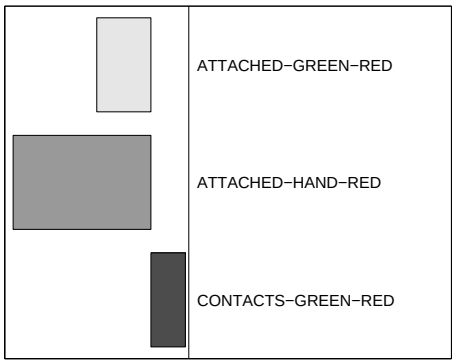


Figure 9.16: Example instances of the frequent Allen pattern F14 found in *put-down* scenes of the Video data. The frequency of visually similar patterns with different relations due to slightly shifted interval boundaries are listed in Table 9.8.

observed, then *attached-green-red* is valid simultaneously when the red block is placed on the green block and finally the result is *contacts-green-red*, without the hand. The end points of *attached-hand-red* and *attached-green-red* are aligned and occur exactly one time unit before the *contacts-green-red* interval starts. This corresponds to the *finished by* and *meets* relations, respectively. A closer look at the data revealed several infrequent fragments of this pattern. Small shifts in the interval end points change the relations, the occurrences of all possible combinations is listed in Table 9.8. Only one combination is frequent enough to appear in the mining result in form of pattern F14. The remaining 12 scenarios of *put-down* are fragmented into four different rare patterns.

We did not expect to find patterns for the *move* scenes, because there are only 15 repetitions. Nevertheless, some patterns are found because they also appear in some of the more complex scenes. All patterns for *move*, *(un)stack*, and *(dis)assemble* only describe fragments of the respective experimental setups, because they are simply too small. Using only three intervals, the patterns for *move* recognize only one of the two stacks of two blocks present at the beginning and the end of the scenes. Similarly, in the *(un)stack* and *(dis)assemble* scenes the simultaneous presence of three intervals (the hand touching a stack of three blocks) is never discovered. As an example we show pattern F20 found in 29 out of 30 of the *assemble* scenes in

		attached-hand-red and *attached-green-red*		
		during	finished by	overlaps
attached-hand-red	overlaps	6	0	0
and	meets	2	18	0
contacts-green-red	before	0	1	3

Table 9.8: Occurrences of fragments of Allen patterns in the *put-down* scenes by considering similar alternatives for the relations between the intervals.

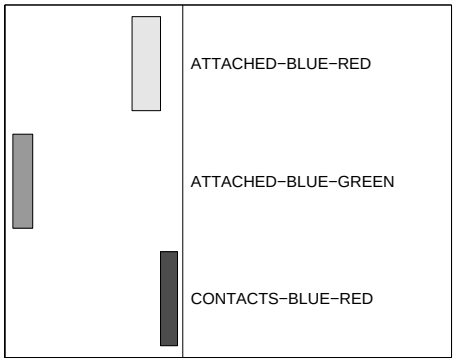


Figure 9.17: Example instances the frequent Allen pattern F20 found in *assemble* scenes of the Video data. Only fragments of this complex scene are explained by the pattern.

Figure 9.17. It merely consist of the sequential occurrence of *attached-blue-green*, *attached-blue-red*, and *contacts-blue-red*, not even remotely explaining the complex scenario involving three blocks and the hand.

9.4 Unification-based Temporal Grammar

One major difference between the TSKR pattern language and its ancestor UTG is the extension of Event patterns from almost synchronicity of a fixed number of intervals to the arbitrary sized Chord patterns expressing the more general temporal concept of coincidence. In Chapter 7.3 we have shown how this leads to a higher temporal expressivity. We briefly demonstrate the effects of restricting the search space to synchronous Event patterns on the two interval data sets.

The mining of Events was performed in two steps. We first mined Chord patterns with a minimum size according to the definition of Events and then pruned all instances that could not be considered almost synchronous in a post-processing step. Let s_f and s_l be the first and last start points and e_f and e_l be the first and last end points of all participating Tone intervals (see Figure 9.18 for an example). If $\frac{s_l - s_f + 1}{e_f - s_l + 1} > \epsilon$ or $\frac{e_l - e_f + 1}{e_f - s_l + 1} > \epsilon$ for a threshold ϵ determining the strictness of synchronicity, the Chord instance was discarded.

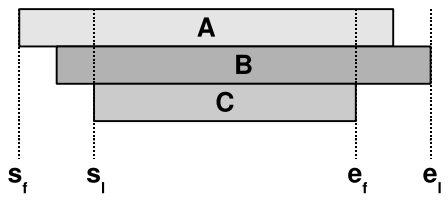


Figure 9.18: The first and last start and end points are used to check the almost synchronicity of a Chord to obtain UTG Events from Chords.

9.4.1 Skating data

The initial Chord mining was performed with the same parameters as in Chapter 9.2.1, except for the minimum size 5 to enforce Event patterns providing state descriptions for all Aspects. A total of five Chords was found. Four of these Chords had been selected and labeled by the expert in Chapter 9.2.1. The effect of pruning the instances of each Chord with different values of the synchronicity threshold to obtain Events is shown in Figure 9.19. As a reference, the number of Chords is shown on the right, corresponding to no synchronicity restrictions ($\epsilon = \infty$). Up to values of $\epsilon = 0.4$ virtually no Event patterns are found. Note, that this already corresponds to a very relaxed notion of synchronicity. The maximum temporal difference between the interval start points (and end points, respectively) is allowed to be 40% of the duration of the core part of the pattern where all intervals are valid simultaneously. Even for values up to $\epsilon = 1.0$ none of the interesting patterns is found as a frequent Event with a minimum frequency threshold of 8. Only the *swing* pattern appears as an Event with about 25% of the occurrences of the corresponding Chord. The Event labeled *unknown* is the only pattern that would have exceeded the minimum frequency with $\epsilon > 0.5$. It was not selected by the expert in Chapter 9.2.1, however, because it appears only on 3% of the time points and does not represent an interesting combination of Tone patterns. Further processing to find higher level patterns expressing order was not possible with only one Event pattern.

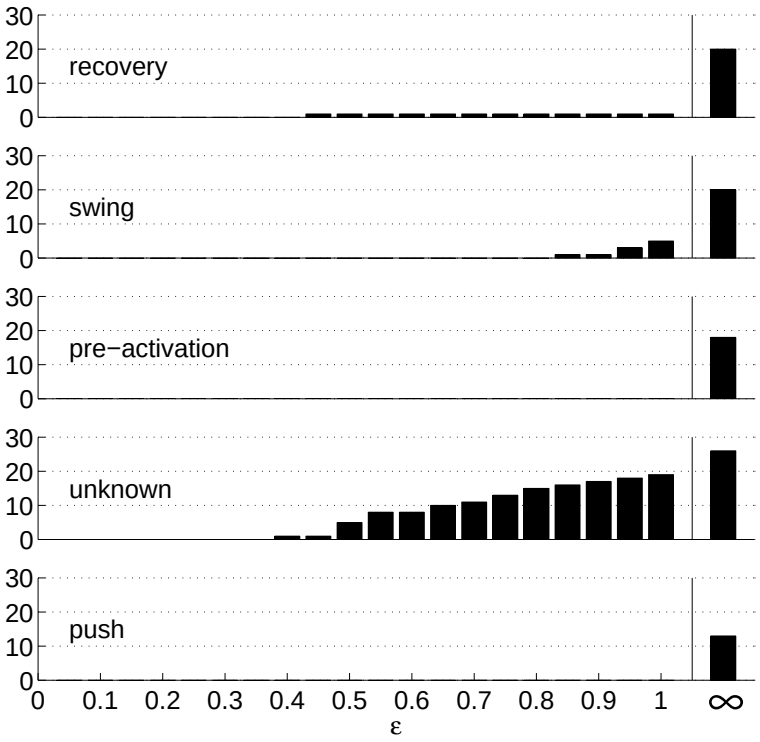


Figure 9.19: The Frequency of UTG Events for different values of ϵ found in the Skating data. The frequency of the unrestricted 5-Chords found with TSKM is shown on the right hand side. The smaller the parameter ϵ, the more instances are dropped because they are not considered almost synchronous. Only one Event is found for reasonably small values of ϵ.

9.4.2 Video data

The Video data does not provide the notion of Aspects. For the mining of TSKR patterns we resorted to eight Aspects with a single state each, but it would not make sense to require Events to contain all eight states simultaneously. There would only be one possible Event that never occurs. We therefore graciously allow Events to be of arbitrary size on this data set and only raised the minimum size of the Chords to two intervals to be able to measure the amount of synchronicity. This leads to nine Chord patterns. Figure 9.20 shows the effect of different values of ϵ on the occurrences of the resulting Event patterns in comparison with the count for the unrestricted Chord patterns, the participating Tone intervals for each pattern are listed in Table 9.9

Event	Tones
P1	*contacts-blue-green*, *contacts-blue-red*
P2	*contacts-blue-green*, *contacts-green-red*
P3	*contacts-blue-green*, *attached-hand-red*
P4	*attached-blue-hand*, *attached-blue-green*
P5	*attached-hand-red*, *attached-blue-green*
P6	*attached-hand-red*, *attached-blue-green*, *attached-blue-red*
P7	*attached-hand-red*, *attached-blue-green*, *attached-green-red*
P8	*attached-hand-red*, *attached-blue-red*
P9	*attached-hand-red*, *attached-green-red*

Table 9.9: Tones that appear almost synchronously as UTG Events in the Video.

Again, most patterns have very low frequencies for values as high as $\epsilon = 0.7$. Only the two top most patterns (P1 and P2) are almost as frequent as the unrestricted Chord even for low values of ϵ. Both describe the co-occurrence of two *contacts* relations, i.e., a stack of three blocks. Recall, that while the hand is touching the stack, all blocks in the stack are related by *attached* relations, the two *contacts* relations thus disappear synchronously when the hand touches the top most block and (re)appear when the hand releases it.

Pattern P3 is observed about half as often as the corresponding Chord for $\epsilon > 0.6$. All other patterns are not found for all reasonably low thresholds. They all consist exclusively of *attached* relations including one with the hand holding a block. The *attached* relations involving colored blocks describe the stack being touch by the hand. In these scenarios the hand holding a block is always valid long before or long after the block is put on or taken off a stack, the intervals can thus not be considered almost synchronous.

Trying to find order patterns (Sequences) using the two almost synchronous Event patterns P1 and P2 was not possible, because only at most one of the two occurred within each scenario.

9.5 Hidden Markov Models

Hidden Markov Models (HMM) (Rabiner, 1989) are a very popular method for modelling multivariate time series with underlying states. We have described numerous applications of HMM in Chapter 5. We evaluate how well a HMM can capture the temporal structure of the Skating data and analyze the explanation capabilities of the model. Only the Skating data set is available for this task, because the intervals of the Video data were directly obtained from the videos and not from time series.

We used the 10-dimensional time series for HMM training obtained by merging the five Aspects (see Chapter A.1.2). All time series except the angular displacements were rescaled to the interval $[0, 1]$ to obtain numerical values that can be interpreted as percentages of the

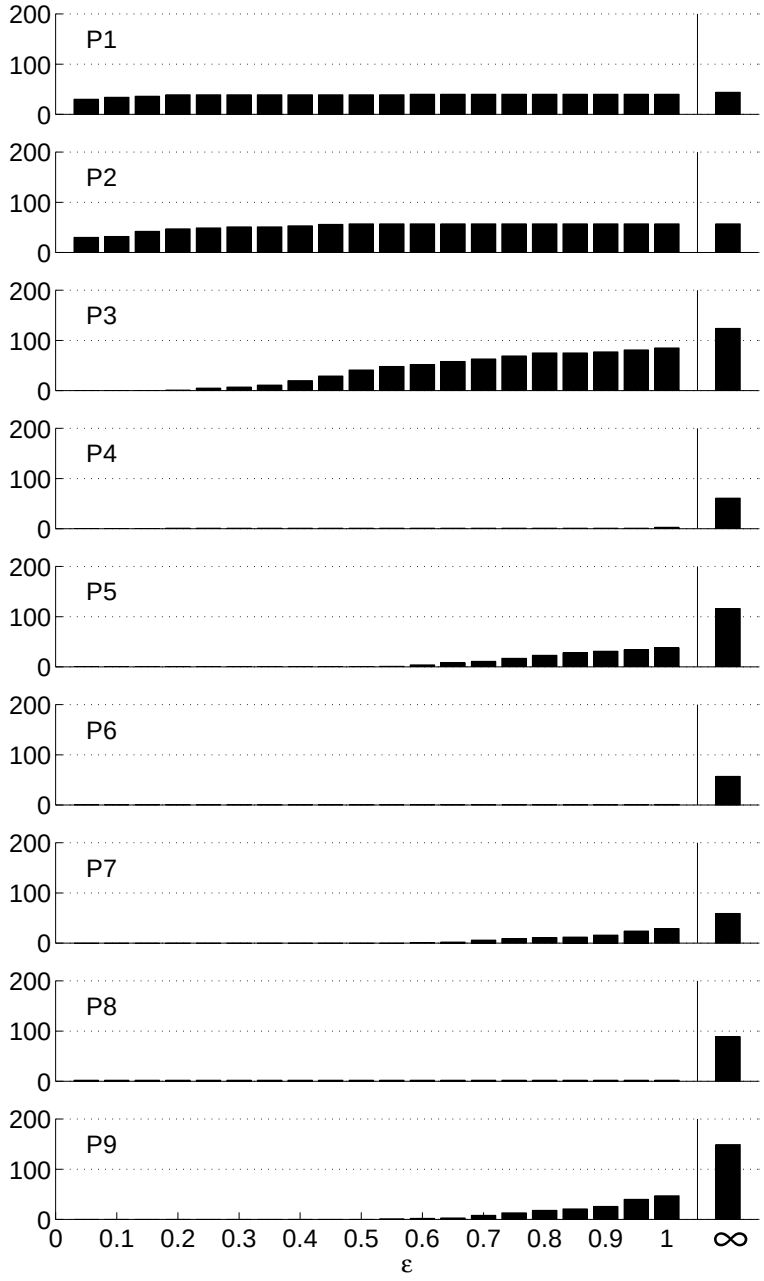


Figure 9.20: The Frequency of UTG Events for different values of ϵ found in the Video data. The frequency of the unrestricted 5-Chords found with TSKM is shown on the right hand side. The smaller the parameter ϵ, the more instances are dropped because they are not considered almost synchronous. Only two Events are found for reasonably small values of ϵ.

maximum. A HMM with Gaussian mixture models (GMM) as output distributions was trained with the HMM Toolbox[4]. The GMM were initialized using k-Means. The maximum number of training iterations was set to 50, the best of ten runs was used.

The aim of building a HMM on the Skating data was interpretation of the underlying phenomena, therefore we tried using 2-20 states. Considering, that the important structure of the data was captured by nine Chords with the TSKM (Chapter 9.2.1) more states do not seem plausible. To enable a one dimensional display of the GMM, a diagonal covariance was used. The evaluation of the models was performed using the Bayes Information Criterion (BIC, Hastie et al. (2001)) that trades of the likelihood of the model given the data and the number of parameters used. In Figure 9.21 the BIC values are shown for HMM with one or two Gaussian distributions per state. There is no clear local maximum within the range of states. For two mixtures, there is a slight decrease from 16 to 17 states, for one mixture, there is no increase from five to six states.

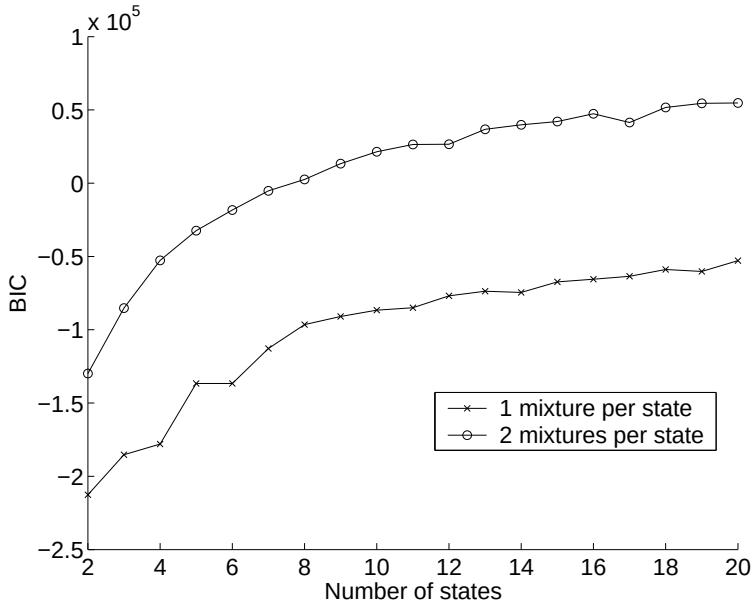


Figure 9.21: BIC scores for HMM models with one or two output distributions per state and several numbers of states trained on the Skating data. No clear maximum is visible within this range, two output distributions are generally better.

The BIC values are in general better for the HMM with two GMM per state. These models sometimes merge two qualitatively different value regions of a variable in a single state. In Figure 9.22 we show the Gaussian output distributions for the Gluteus muscle of the 5-state model with two mixtures. State 1 and 2 have one output distribution in the regions corresponding to low and high muscle activity, respectively. This means that when one of these states is observed no useful information on the Gluteus muscle is given, because it could be either active or not. Similarly, state 4 merges high activation and very high activation.

We therefore used single Gaussian output distribution per state and picked the 5-state model for further analysis. The state sequence obtained by the Viterbi algorithm is shown in Figure 9.23 with the shaded bars below the input time series. A continuously repeating sequence of the five states is clearly visible.

The particular state transition probabilities are shown in Table 9.10. The very high self-transition probabilities on the diagonal indicate typically very long states, the largest values off

[4] `http://www.ai.mit.edu/~murphyk/Software/hmm.html`

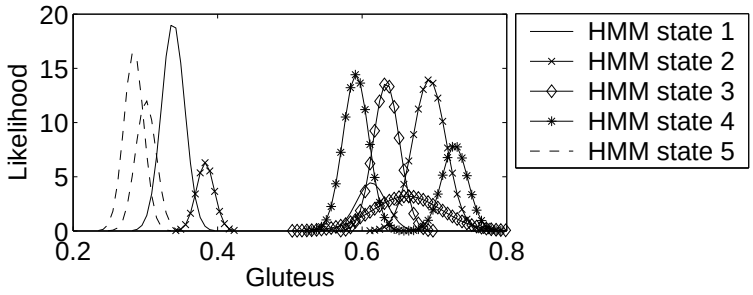


Figure 9.22: Mixture of two Gaussian distributions per state for the Gluteus muscle and a 5-state HMM trained on the Skating data. The mixtures for a single HMM state do not necessarily describe the same qualitative state of the Variable.

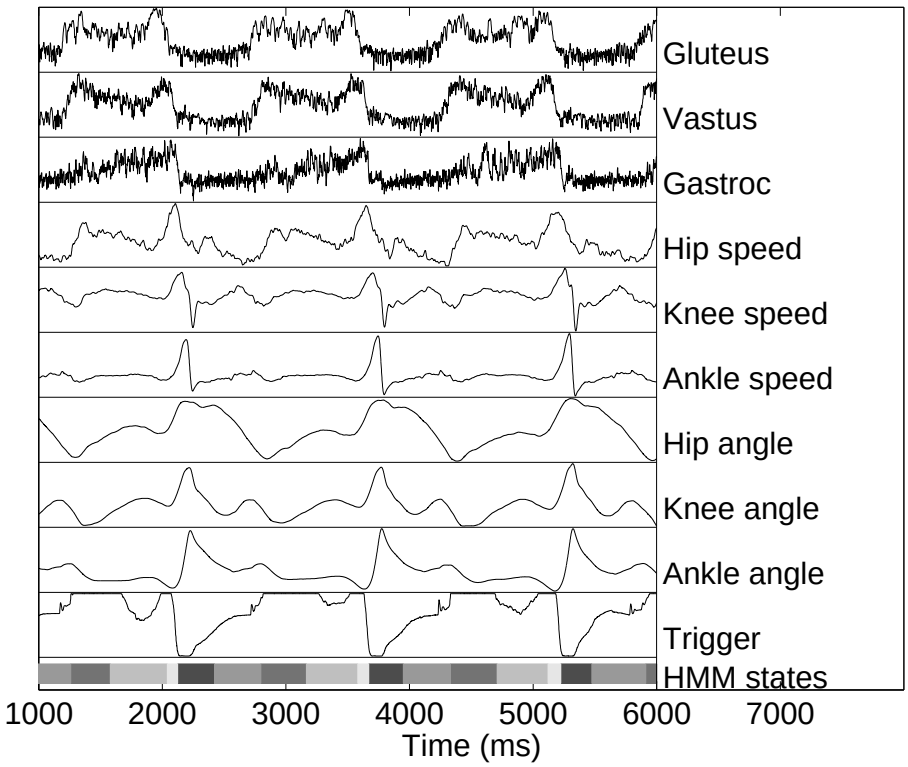


Figure 9.23: Five second excerpt from input time series and the Viterbi state sequence from a 5-state HMM with single Gaussian output distributions per state trained on the Skating data.

the diagonal shown bold indicate likely state transitions. The most typical circular order among the states according to this estimation is 1, 3, 2, 4, 5.

State	1	2	3	4	5
1	0.9965	0.0000	**0.0035**	0.0000	0.0000
2	0.0000	0.9969	0.0000	**0.0030**	0.0002
3	0.0000	**0.0026**	0.9974	0.0000	0.0000
4	0.0000	0.0001	0.0000	0.9976	**0.0023**
5	**0.0104**	0.0000	0.0000	0.0000	0.9896

Table 9.10: Matrix of state transition probabilities for 5-state HMM trained on the Skating data. The off-diagonal row maxima are shown bold.

We analyzed the Gaussian distributions of each state and each variable for possible interpretations. All mean values and standard deviations describing the states are listed in Table 9.11. The mean values within each row can be used to get an impression of the dominant value range in comparison with the other states. For example, state 1 seems to correspond to low pressure of the foot contact (Trigger) and state 2 seems to indicate the foot touching the ground. In particular, for most muscles an assignment of HMM states and qualitative descriptions can be made. In Figure 9.24(a) state 1 and 3 seem to correspond to low activity of the Gastroc muscle, while states 2, 4, and 5 represent successively higher level with few pairwise overlap of the distributions. For the Vastus muscle in Figure 9.24(b) the three levels of activation similar to the analysis in Chapter A.1 are observed. States 1 and 3 correspond to low, 4 and 5 to high, and 2 to very high activity, respectively.

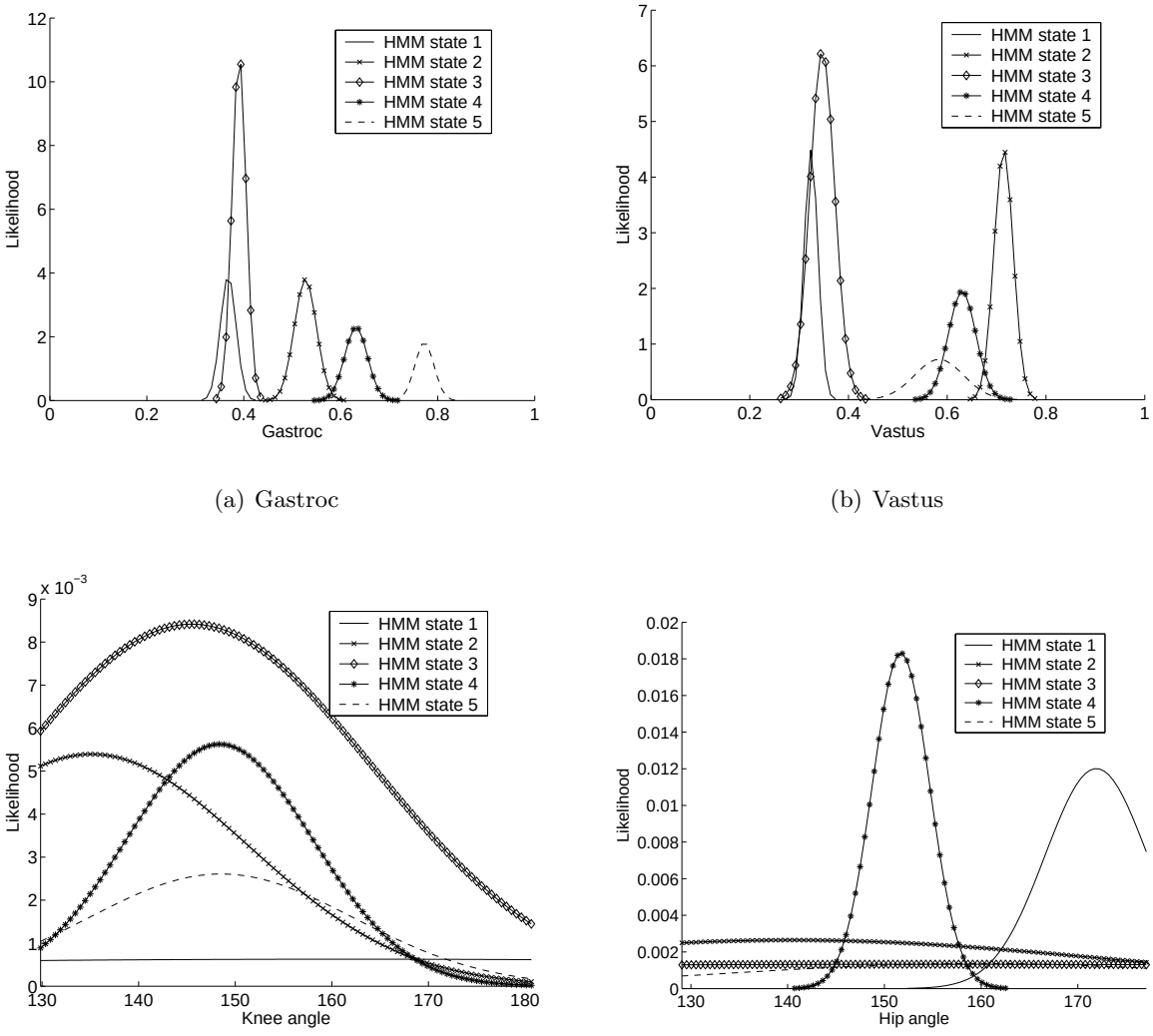
In other cases the output distributions do not offer much explanation, since the output distributions have large overlap and high variance. The standard deviations for many of the angular displacement distributions in particular are very large and do not characterize a distinct state for these variables, e.g., for the knee angle in Figure 9.24(c). For the hip angle two states centered at about 150 and 170 degrees are observed, while the other states show very widespread distributions.

Variable	State 1	State 2	State 3	State 4	State 5
Trigger	0.16±0.03	0.99±0.01	0.64±0.02	0.83±0.03	0.76±0.10
Ankle angle	89.26±159.87	67.08±33.09	76.08±6.13	66.57±5.05	58.16±2.30
Knee angle	161.79±101.25	135.12±16.21	145.50±18.73	148.38±9.65	148.53±13.79
Hip angle	171.91±5.29	140.44±33.04	155.21±118.73	151.71±2.96	159.94±26.26
Ankle speed	0.41±0.07	0.33±0.01	0.36±0.01	0.35±0.01	0.37±0.01
Knee speed	0.54±0.06	0.57±0.02	0.58±0.02	0.60±0.01	0.71±0.02
Hip speed	0.42±0.03	0.52±0.02	0.19±0.02	0.43±0.02	0.83±0.02
Gastroc	0.37±0.02	0.53±0.02	0.39±0.01	0.63±0.02	0.77±0.02
Vastus	0.32±0.01	0.71±0.02	0.35±0.03	0.63±0.03	0.58±0.05
Gluteus	0.29±0.01	0.65±0.02	0.38±0.03	0.68±0.03	0.44±0.02

Table 9.11: Mean values and standard deviation of Gaussian distributions per variable and state for 5-state HMM trained on the Skating data.

9.6 Comparison of Methods

The application of the TSKM to the skating data was highly successful. The two patterns selected from the results offered a good description the cyclical movements at different levels



(a) Gastroc

(b) Vastus

(c) Knee angle

(d) Hip angle

Figure 9.24: Gaussian distributions per state for four variables and a 5-state HMM trained on the Skating data. For the muscle activity (Gastroc and Vastus) the distributions have small variances can be qualitatively described. For the angles (Knee and Hip) much larger variances are observed.

of detail. The next step could be an analysis of such patterns over several skaters and skating speeds to investigate possible intra and inter subject variations. The semantics of the pattern language were found to be easily comprehensible and meaningful within the domain. The usage of the TSKR patterns with the final graphical representation of the Phrases offered a great benefit over simply viewing the input time series, as is common practice in the analysis of such data in sports medicine.

The descriptions found by the TSKM in the Video data in an unsupervised process corresponded exactly to the known experimental setups. The Phrases directly explain the actions without the need to look at the original videos. Reversed patterns were found for reversed scenarios. A possible application of the found patterns is classification. One could mine the TSKR patterns on part of the data and classify the remaining scenes by occurrence of the selected Phrases. It is obvious from Table 9.3 that this would give excellent results on the data set.

The relation of muscle activity states to movement states, all of which are subject to noise from the recording process and the discretizations procedure, does not seem to be well representable with Allen's relations. The patterns mined with Allen's relations from the Skating data were not useful to the expert. For patterns involving many intervals the textual representation was very long and hard to comprehend. The patterns were therefore presented graphically with an example instance from the data. Looking at a pattern instance is almost equivalent, however, to looking at the input data before even mining patterns. The only benefit is that the intervals somewhat represent the most dominant structure of the 19 movement cycles. In order to avoid the selection of an ambiguous pattern one has to check all instances of a pattern, though. A detailed analysis of the relation between the two major leg muscles and the foot contact revealed, that the pattern representation is susceptible to both missing intervals and slightly shifted end points that cause pattern fragmentation. No consistent relationship of these muscles and the foot contact was expressible with Allen's relations.

In the Video data several valid patterns based on Allen's relations were found. The relative positions of the involved intervals corresponded to the known actions in the videos, but most patterns explained only fragments of the scenarios. The most frequent Allen patterns did not show as good a correspondence to the known scenarios as the TSKR patterns (compare Table 9.7 and Table 9.3). As for the Skating data, we did not find the long listings of pairwise relations useful for interpretation. The graphical representation of two similar patterns in Figure 9.13 was used for comparison instead of the textual representations in Figure 9.11 and Figure 9.12. A detailed analysis of the strongest pattern for the *pick-up* scene revealed another example of pattern fragmentation. The most frequent pattern of the *pick-up* scenes was not recognized in more than a third of the examples due to slightly shifted interval boundaries.

We further demonstrated, that the restriction to almost synchronous intervals as performed by the Event patterns of the UTG is too harsh on both data sets. For all sensible values of the threshold determining almost synchronicity hardly any patterns were found. Even when stretching the concept of synchronicity with quite large threshold values, only one Event pattern with low temporal support was found in the Skating data making it impossible to mine Sequence patterns. In the Video data, two Events were found and recognized as valid examples for synchronous temporal phenomena. In some applications the distinction between general Chords and Events (i.e. almost synchronous Chords) might be useful. In the Video data, however, many other important Chords that composed the Phrase patterns found in Chapter 9.2.2 are not recognized as Events and no Sequence patterns involving the two remaining Events were found.

The HMM for the Skating data was able to capture the circular movements with a consistent sequence of states. There was no clear indication on the number of states, however, the five state model was chosen rather arbitrarily. Without further processing the HMM does not offer an easily comprehensible description of these states. Even for only five states the model consist of

135 numerical parameters. In order to interpret these parameters, knowledge on the structure of HMM is needed. One needs to be familiar with the concept of transition probabilities, as well as Gaussian distributions in order to understand the tabular and graphical representations given in Chapter 9.5. Gaussian distributions with very large variances can be seen as an indicator, that this variable is not very characteristic for a state, but in principle the likelihood of each state always depends on all variables, whether they are relevant or not. In contrast to the rule based models, the HMM does not offer abstraction mechanisms over the numerical values. Further, it is designed to model the complete time series, whereas the other methods list patterns that describe frequently occuring local phenomena.

Chapter 10

Discussion

The aim of this thesis is to develop effective means for the description of local temporal phenomena in multivariate time series. In order to use the results for knowledge discovery we required them to be understandable to humans. For the matter of understandability we distinguish symbolic and sub-symbolic methods (Ultsch and Korus, 1995).

Sub-symbolic methods model the structure of data using many numerical parameters. They are usually aimed at prediction or classification. The output of sub-symbolic methods often depends on the values and interactions of most or all model parameters. They fail to explain the prediction or the classification. There are certainly areas of data mining where it is sufficient to build such black-box models that can approximately reproduce a classification or predict future data. An important requirement for knowledge discovery is the interpretability of the results. In many domains the expert wants to know why a decision was made or what a frequently occuring pattern describes. Comprehensible descriptions of the models are crucial for the success in this case.

As an example, consider the sub-symbolic method of Hidden Markov Models (HMM) (Rabiner, 1989) applied to the Skating data. The HMM succeeded in capturing and reproducing the temporal structure of the time series, but it failed to provide understandable explanations. For this application the prediction of future behavior is useless. We did know that the person repeats the same movement cycle, as it was an experiment under controlled conditions. Instead we were interested in dominant patterns that explain the relations of the variables.

Symbolic methods are based on discrete data models. This provides the basis for understandability, as each data item can represent a qualitative description of small parts of the data. We think that only symbolic representations can be easily understood by humans, especially if they are not familiar with the complicated mathematical models used by many sub-symbolic methods. However, models built with symbolic data are not necessarily understandable. If too many or too large of patterns are produced, the human interpretation is hindered or made impossible. We therefore further distinguish approaches aimed at automated data processing or interactive knowledge discovery. For automated data processing methods, the results are not meant to be analyzed manually, but provide input for other computer programs. In this case the symbolic data is mainly of advantage for efficiency. Interactive knowledge discovery depends on human analysis of the usefulness and novelty of the mined patterns. It is of utmost importance, that neither the size of the patterns, nor the amount of patterns overwhelms the user. The analyst will simple be lost and is less likely to continue the cooperation in the mining endeavor. Successful data mining strives upon the communication between the experts in pattern mining and the experts of the application domain. The former are needed to apply advanced techniques and discover potentially interesting structures in the data, the latter are needed to evaluate and utilize the results. We see our work in the scope of such symbolic methods aimed at human interpretation.

We reviewed and analyzed many existing symbolic approaches for unsupervised mining of temporal rules from time series data. Time point based methods were not found to be well equipped for our purpose, because the important temporal concept of duration cannot be expressed by most methods. Interval based methods seemed more promising because duration is modeled by the lengths of the intervals. The most powerful approaches for mining symbolic interval patterns were carefully analyzed, namely the Unification-based Temporal Grammar (UTG, Ultsch (1996b)) and patterns expressed with Allen's relations (Allen, 1983; Höppner, 2003).

Based on the findings we designed a new representation that is inspired by the advantages of both the UTG and Allen's relations and compensates the identified problems. The TSKR pattern language combines the high temporal expressivity of Allen's relations with the robust matching of interval boundaries of the UTG. By transferring the concept of partial order from the time point based Episode (Mannila et al., 1995) patterns to interval data, a temporal expressivity even beyond the patterns using Allen's relations is obtained.

The hierarchical representation and the consistent use of semiotic triples for pattern representation, both inherited from the UTG, are highly promising for interactive knowledge discovery tasks. Each pattern element describes a single temporal concept and is thus better comprehensible as a mixture of multiple concepts as represented by many of Allen's relations. The hierarchical composition of such elements to larger patterns offers a high level of abstraction producing a compact representation that provides details on demand. The semiotic triples offer an annotation of semantical and syntactical information with pragmatic descriptions to enable better interpretation on higher, more abstract levels. The separate mining of single temporal concepts leads to smaller search spaces and enables human interaction in analysis and filtering of (partial) results.

The extensions of the UTG to the temporal concepts of coincidence and partial order posed new algorithmic challenges. Similar to patterns mined with Allen's relations we utilized well-known methods from itemset mining. Both the mining of coincidence and of partial order were formulated as frequent itemset problems, enabling the use of established efficient algorithms and lattice-based visualization of pattern sets. The novel concept of margin-closedness effectively prunes the results so they can be manually analyzed. From a set of similar patterns with almost the same frequency, only the most specific pattern is retained. The amount of pruning is directly controlled by a single parameter that ensures that all reported patterns have a minimum relative difference in frequency. Rare specific as well as frequent general patterns are present in the result. The pruning of the search space directly affects the runtime of the mining. Few patterns can be found very fast and the parameter can be changed if the results are too coarse.

The detailed evaluation of the TSKM on two data sets showed the applicability and usefulness of the new pattern language, the mining algorithms, and the graphical representations of the results. The theoretical advantages in expressivity, robustness, and interpretability over other interval rule languages were observed to be relevant in practice.

More insights on the properties of the TSKR and the mining algorithms could be obtained by analyzing more datasets with different characteristics and with artificially generated data. A preliminary analysis of several additional multivariate time series showed promising results. The TSKR is only applicable, however, to multivariate time series that contain persisting states. Only if some properties of the time series are observable during time intervals of a certain minimum length (w.r.t. the time resolution of the measurements) we can extract Tones, Chords, and Phrases. Chaotic time series as observed in some physical systems are probably not well representable by the TSKR. For the TSKM algorithms we further require that the temporal phenomena are repeatedly observed, as the most frequent patterns are searched.

In the remainder of this chapter we discuss the criteria we used to compare the TSKR to other pattern languages and our choices of mining algorithms. In particular we reflect upon

interpretability according to the Gricean maxims (Grice, 1989). Existing and possible extensions of the alternative interval pattern languages to increase the expressivity and robustness are mentioned. We conclude with a comparison of our novel concept of margin-closedness to existing approaches for condensed itemset representations and give hints to future work.

We are claiming higher interpretability for the TSKR pattern language in comparison with patterns using Allen's relations. For the Skating data an expert confirmed the results to be easily comprehensible and more useful than the patterns expressed with Allen's relations. For the Video data set the patterns clearly described the known temporal phenomena in the data whereas Allen's relation captured only parts thereof. The concept of almost synchronicity of the UTG turned out to be too restrictive for both data sets. The TSKM has further produced promising results for many additional data sets from various domains including stock market data, music (Mörchen and Ultsch, 2004), robot sensors and Australian sign language (Hettich and Bay, 1999) and cell phone sensors (Mäntyjärvi et al., 2004).

We used the Gricean maxims listed in Figure 10.1[1] to compare the interpretability of interval pattern languages. This was suggested by Sripada et al. in a study where natural language descriptions of time series were generated and the interpretability was evaluated by experts in three different application domains (Sripada et al., 2003b). The authors noticed that the suggested changes to the initial knowledge representation followed the Gricean Maxims. In our comparison, the maxims of relevance and quantity as well as the aspect of ambiguity from the maxim of manner revealed major differences between the pattern languages.

- Maxim of Quality: Truth
  - Do not say what you believe to be false.
  - Do not say that for which you lack adequate evidence.
- Maxim of Quantity: Information
  - Make your contribution as informative as is required for the current purposes of the exchange.
  - Do not make your contribution more informative than is required.
- Maxim of Relation: Relevance
  - Be relevant.
- Maxim of Manner: Clarity
  - Avoid obscurity of expression.
  - Avoid ambiguity.
  - Be brief (avoid unnecessary prolixity).
  - Be orderly.

Figure 10.1: The Gricean maxims of conversation (Grice, 1989) can be seen *"as presumptions about utterances [...] that we as listeners rely on and as speakers exploit"* (Bach, 2005).

The maxim of relevance is well supported by the hierarchical structure of the TSKR inherited from the UTG. An expert can provide feedback on which patterns are relevant and interesting to

[1] Taken from `http://en.wikipedia.org/wiki/Gricean_maxims`

him and discard valid but known patterns before the next level constructs are searched. Due to the higher expressivity of the TSKR compared to the UTG, generally more patterns are found. The usage of margin-closedness provides a powerful mechanism to reduce the size of the result and produce a concise representation of the most important patterns supporting the maxim of quantity.

Allen's relations were shown to be prone to producing ambiguous patters, i.e., the same rule describing quite different pattern instances. This could be prevented by splitting patterns with potential high variability into several different patterns, e.g., using relations involving the interval mid points (Roddick and Mooney, 2005). However, using 49 instead of 13 relations with many additional conditions requiring equality of time points will in turn increase effects of pattern fragmentation. Note that all pattern languages are vulnerable to ambiguity caused by uniform warping, e.g., Chords involving the same coincidence of intervals but occuring with widely differing durations. When observed, this can be avoided by using additional constraints on minimal and maximum pattern duration.

In general it can be argued, whether the Gricean conversational maxims are good guidelines for the design of knowledge representations in data mining. The interpretation of mining results is not a conversation of humans, as many additional ways of expressing the meaning of language are not available. The maxims are formulated as prescriptions, but according to Bach they are *"better construed as presumptions about utterances, presumptions that we as listeners rely on and as speakers exploit"* (Bach, 2005). We think that the output of data mining programs that is targeted at humans for analysis, should take these presumptions into account in order to avoid misinterpretation that cannot be recognized or corrected by the program. It is of particular importance to report only true results according to the maxim of quality and to report only a manageable amount of patterns according to the maxim of quantity, in order not to mislead or overwhelm the user.

Less philosophical arguments for the comparison of interpretability of the pattern languages could be drawn from user questionnaires (e.g. Guimarães (1998)) or psychological tests. This is beyond the scope of this thesis. The participants of such a test could be asked to describe given interval patterns with their own words and to draw examples of patterns described with Allen's relations or the TSKR. Imagine being given the listing of 15 pairwise Allen relations in Figure 7.21 and the task to draw a qualitative example of the pattern. This is not at all a trivial task, many dependencies need to be considered. In fact, this corresponds to the constraint satisfaction problem of temporal reasoning which is NP-complete (Vilain et al., 1989). We think, that the same task is rather easy given a description in the TSKR language and shouldn't this be the case in order to call a pattern understandable?

We proposed a novel measure for the robustness of interval patterns given noisy interval boundaries. Only simple patterns were analyzed, as the main purpose of the measure was to point out the conceptual differences of the pattern languages. The breakdown values only consider shifted interval end points, other types of noise such as interruptions of intervals or missing intervals are not accounted for. Extensions of the pattern languages to allow alternative labels for positions in the pattern or to allow a certain amount of errors (Pei et al.; Yang et al., 2001a) can be used to deal with these aspects of noise. The UTG already uses disjunctions on the most abstract pattern level, called Temporal Patterns. For Allen's relations, disjunctions of implication rules (Höppner, 2002a, 2003) and disjunctions of intervals within a rule (Höppner, 2002b, 2003) have been proposed. These method exploit the information content of rules, however, and cannot be used with the basic interval patterns. For the TSKR we omitted such formulations for clarity, the integration of disjunctions, negations, or error-tolerance into the pattern language and the mining algorithms offers room for future explorations.

The dependence on exactly equal interval endpoints in many of Allen's relations is the reason for missing robustness on noisy interval data. There are certainly cases when the exact start and

end points of intervals matter. For example, in marketing it might be of interest, if consumption went up during or after an advertising campaign started. In medicine the exact relation of treatment periods and different health states of the patient can lead to insights about influence of the medication. We think, however, that the use of Allen's relations with inexact data is conceptually flawed. Here, no distinction should be made between intervals starting or ending exactly at the same time or within a small time window. Ignoring this fact leads to fragmented patterns. Introducing thresholds for a more robust notion of equality of interval boundaries does not completely solve the problem. The threshold needs to be chosen carefully to distinguish, e.g., meeting intervals from overlapping intervals. The fragmentation problem is merely shifted to variations of interval boundary differences slightly below or above the threshold value.

The problem of fragmented patterns has been recognized by Höppner for rules not contained in the search space (Höppner, 2002a, 2003), but not for the more simple cases with interval patterns we identified. As a solution, similar rules are merged using disjunctions but this makes the patterns even longer than they already are for all but very simple cases. Note that such large patterns were also observed in practice.

The lower expressivity of UTG Events in comparison to TSKR Chords was disastrous in the two applications shown. The idea of synchronous intervals may be of interest in other applications, though. In these cases it is easy to filter the more general Chords for approximate synchronicity in a post-processing step to obtain the equivalent of UTG Events. The reason for the TSKR being more expressive than Allen's relation is the usage of partial order. In principle it is possible to mine partially related patterns with Allen's relations by allowing blanks in the matrix of pairwise relations. We did not pursue this idea, because of the other disadvantages w.r.t. robustness and interpretability.

The proposed algorithms for the TSKM provide an efficient methodology to mine TSKR patterns based on established itemset mining techniques. It might be possible to further improve the efficiency by utilizing the special structure of the data, namely itemset intervals and items representing closed sequential patterns.

The problem of mining Chords and Phrases is quite similar to the well known sequential pattern mining from itemset sequences. Each time point of a symbolic interval sequence could be considered a transaction with an itemset holding one item for each symbolic interval that contains the time point. The method of (Casas-Garriga, 2005) can be used to mine partial orders of itemsets resulting in patterns similar in structure to our combination of Phrase and Chord patterns. This way the selection of a subset of the intervals and the mining of order will be done simultaneously. We think, that our methodology offers significant advantages. With time series containing persisting states the long interval durations will cause a large waste of computing resources. For the Skating data one transaction contains about 1,500 time point itemsets with mostly five items and many exactly repeating itemsets. Using time interval data models that represent states with duration largely reduces this complexity. Using the data model conversion described in Chapter 8.5.1 directly on the interval sequence of Tones leads to much fewer itemsets per transaction (for example about 15 itemsets for the Skating data). This still turned out to be a very dense data set. The PrefixSpan (Pei et al., 2001) algorithm for sequential pattern search did not terminate after more than one million function calls, whereas the search using the interval itemset sequence of the nine selected margin-closed Chords finished with less than a thousand calls.

The complexity of searching frequent itemsets is exponential in the total number of items. For Chord mining this means that there might be performance problems with very high-dimensional time series, for Phrase mining the complexity grows with the number of closed sequential patterns. Searching margin-closed patterns is an effective way to control the complexity of the search space. Nevertheless, there might be datasets where the TSKM algorithms become infea-

sible. It would be interesting to check, however, whether the number of Chords really grows at an exponential rate for high dimensional data sets.

The separate mining of coincidence and the subsequent restriction to single picks from itemsets of Chords in Phrase mining seems to largely reduce the problem complexity. Further, it offers the possibility to interpret and filter the Chord patterns and revise parameter choices before advancing to the next level. This way observations on the temporal structures of duration and coincidence in the data as well as prior knowledge can be integrated in the mining process. Using the pattern lattice (Zaki and Hsiao, 2005) or other itemset visualizations (e.g. Steinbach et al. (2004)) the remarkable power of human perception (Shneiderman, 1996) can be put to use. Recently, similar approaches integrating human interaction in data mining have been suggested for classification (Ankerst et al., 2000) and clustering (Aggarwal, 2001).

The concept of margin-closedness was very valuable in the evaluation of the TSKM. It largely reduced the amount of reported patterns, while preserving a variety of patterns ranging from small patterns with high support to large patterns with low support. To the best of our knowledge it represents a novel way of lossy compression for itemsets. There are numerous methods available that provide condensed itemset representations, but they have mostly been developed with different aims.

One common motivation is to find a small representation from which the support of all itemsets can be exactly (Bykowski and Rigotti, 2001; Kryszkiewicz, 2001; Kryszkiewicz and Gajek, 2002; Calders and Goethals, 2002, 2003) or approximately (Boulicaut and Bykowski, 2000; Pei et al., 2002; Boulicaut et al., 2003) derived. This makes frequent itemset mining feasible on dense data sets where mining all frequent sets is prohibitively expensive. Association rules can be generated based upon the condensed representation. The basic idea behind all these approaches is to derive the support of an itemset given the support of all subsets (Calders and Goethals, 2003). If this is possible, the larger set does not need to be stored in the result. In TSKM we are particularly interested in large itemsets, corresponding to specific patterns. It would be very inconvenient to only present the list of smaller non-derivable patterns (Calders and Goethals, 2003) to the user and have him carry the burden of deriving the support for larger patterns.

Another line of work is motivated by the fact, that transaction data is often noisy. The strict definition of support, requiring all items of an itemset to be present in a transaction, is relaxed. Early approaches (Boulicaut and Bykowski, 2000; Pei et al.; Yang et al., 2001a) suffered mainly from two problems (Seppänen and Mannila, 2004): smaller patterns could be less frequent than larger patterns and the same item could be missing from all involved transactions. These problems were fixed by the definition of dense itemsets in (Seppänen and Mannila, 2004). Though not directly designed for condensed representation, the experimental results are shown to be of moderate size if the parameters are chosen accordingly. More directly aimed at summarizing a collections of itemsets with a smaller number of representative sets are (Han et al., 2002; Afrati et al., 2004; Yan et al., 2005). In (Han et al., 2002) only the most frequent, not the most specific itemsets are reported. The motivation of (Yan et al., 2005) is quite similar to our notion of margin-closedness, as it is noted that *"a user may not only be interested in a small set of patterns, but also patterns that are significantly different"* (Yan et al., 2005). All these approaches for approximate itemset mining have one thing in common - they report false information and thus violate the Gricean maxim of quality. They only report approximate support values and possibly list itemsets that are not present in the collection at all (Afrati et al., 2004) or with much smaller support. For the TSKM we rather present true information, i.e., exact patterns with exact support. We simply reduce the amount of patterns reported preserving the variety of the collection.

Most similar to margin-closedness are the generalized closed itemsets (Pudi and Haritsa, 2003). All supersets with approximately the same support as a smaller itemset are pruned. In

contrast, we prune the smaller subsets with support similar to a larger, more specific itemset. This favors more detailed, thus generally more interesting patterns.

As the main challenges for future work we see the development of new interestingness measures and mining algorithms for the TSKR patterns. The proposed algorithms return a concise representation of the most frequent patterns. The pattern language is not restricted to this frequentist view of data mining.

For unsupervised mining measures of surprisingness could be developed in order to search for the most unusual instead of the most frequent patterns. For Chords the conditional probability could be used (Guimarães, 1998), for Phrases recent results from time point data models (Gwadera et al., 2005) could be transferred to time interval data.

Similar to association rules in general and temporal rules in particular (Harms et al., 2002; Höppner, 2003), implication rules could be generated from TSKR patterns. This calls for mining algorithms that directly optimize interestingness measures for rules (Hilderman and Hamilton, 1999), e.g., genetic programming (Saetrom and Hetland, 2003). Such rules could be used for the prediction of qualitative states offering not only some measure of confidence in the implication rule but also an explanation of the current and future temporal phenomena involved. Techniques for rule generalization (Höppner, 2003) could be used to find disjunctive rules.

An important unsolved problem in time series data mining is the construction of understandable and reliable models for the classification of time series. Existing methods often fail to explain the decision on a particular instance. In many application it is of utmost importance, however, to present high level semantic description of the classes to the people that have collected the data. Very fast or very accurate methods are of little merit, if the decision makers do not understand them and do not base their actions on the results.

In the supervised mining context the most characteristic or discriminative patterns for a class of time series could be searched. Based on probabilistic models of pattern likelihood, established techniques like EM training (Dempster et al., 1977) or Gibbs sampling (Casella and George, 1992) are available for model building. Other algorithmic paradigms, like inductive logic programming (Lavrac and Dzeroski, 1994), could also be used to mine the patterns from positive examples only (Fern, 2004).

Recently, interest in data stream mining has been increasing. Similar to early time series data mining research, few emphasis is put on understandability of the results. It would be interesting to develop efficient methods for mining TSKR patterns from streaming time series.

Chapter 11

Summary

In this thesis we presented a novel way of extracting understandable patterns from multivariate temporal data with the aim of knowledge discovery. We defined the Time Series Knowledge Representation (TSKR), a new pattern language for expressing temporal knowledge. Efficient algorithms for mining such patterns were presented, completing our proposed method called Time Series Knowledge Mining (TSKM).

After an introduction to data mining and knowledge discovery in general and temporal data mining in particular we presented an extensive survey of time series data mining. According to our taxonomy of the field, this work falls into the category of unsupervised rule mining. We reviewed existing rule mining methods for symbolic time point and time interval data models. Two powerful existing approaches, the Unification-based Temporal Grammar (UTG) (Ultsch, 1996b) and Allen's relations (Allen, 1983; Höppner, 2003) served as motivating examples in the design of the TSKR pattern language.

Numerical data is first converted into qualitative descriptions of local structure represented as intervals. Typical temporal relations of the intervals are extracted as patterns with a representation close to human language. The patterns can express the temporal concepts of duration, coincidence, and partial order.

The new representation of temporal knowledge was shown to have advantages in interpretability, expressivity, and robustness. Interpretability in particular was thoroughly analyzed according to the Gricean maxims (Grice, 1989). The maxim of quality states that only well supported facts and no false descriptions should be reported. The maxim of relevance requires, that only patterns relevant to the expert are listed. The maxim of manner requires communication to be brief and orderly and to avoid obscurity and ambiguity. The maxim of quantity states that neither too much nor too few should be reported. According to these criteria many theoretical advantages of the TSKR were shown. High interpretability is achieved by the hierarchical structure of the patterns and the discovery process, consistent use of annotations, and avoiding redundancy in the results.

The TSKR patterns can be efficiently mined with methods adapted from itemset mining. The novel concept of margin-closedness enables the creation of concise pattern sets that explain various aspects ot the temporal data ranging from more frequent and less specific to less frequent but more detailed descriptions.

We have further demonstrated the superiority and usefulness of the TSKM by presenting two detailed practical examples from different application domains. In an application to sports medicine the results were recognized as valid and useful by an expert of the field. In the other example the patterns discovered from qualitative descriptions of actions observed in videos exactly described the corresponding experimental setups. More promising results have been obtained on a variety of data sets including stock market data, music (Mörchen and Ultsch,

2004), robot sensors and Australian sign language (Hettich and Bay, 1999) and cell phone sensors (Mäntyjärvi et al., 2004).

Our methods are very general and will most likely be useful in other domains and for other data mining tasks like classification or prediction. We believe that these areas could profit from our highly understandable representation of temporal knowledge.

Appendix A

Datasets

A.1 Skating Data: Muscle Coordination during Inline-Speed-Skating

In this chapter we describe the preprocessing of the skating data introduced in Chapter 9.1. Originally, six athletes performed standardized indoor test of graded speed levels at 6 different speeds (3.72, 4.56, 5.39, 6.22, 7.05, 7.89, 8.72 m/s) on a large motor driven treadmill. On each speed level electromyography (EMG) and kinematic data were measured for 30 seconds, corresponding to 12-20 movement cycles depending on selected athlete and skating speed. For this study we selected a single skater running at 7,89m/s performing 19 movement cycles. We collected data from professional athletes, because highly trained individuals show less variation among the single cycles.

The EMG data of 7 major leg extensor and flexor muscles of the right body side were collected using bipolar surface electrodes attached to the relevant muscles. EMG signals were pre-amplified with a gain of 2500 and band-pass filtered (low cut-off: 10 Hz, high cut-off: 700 Hz). Lower extremity kinematics were recorded with 3 angle sensors (electrogoniometers) attached to the ankle, the knee, and the hip. An inertia switch produced a time series indicating the first ground contact. All sensors were sampled at a rate of 1kHz. Table A.1 summarizes the input data. The short names in the first column will be used in the rest of this Chapter.

Name	Description
Trigger	Pressure sensor determining foot contact.
Ankle	Displacement angle of ankle joint.
Knee	Displacement angle of knee joint.
Hip	Displacement angle of hip joint.
Tibial	EMG of Tibialis Anterior (muscle of the anterior compartment of the lower leg).
Gastroc	EMG of Medial Gastrocnemius (posterior part of calf muscle).
Vastus	EMG of Vastus Medialis (medial part of quadriceps femoris).
Rectus	EMG of Rectus Femoris (middle part of quadriceps femoris).
Semiten	EMG of Semitendinosus (medial part of hamstrings).
Gluteus	EMG of Gluteus Maximus (large part of buttock muscle).
Tensor	EMG of Tensor Fasciae Latae (muscle of the anterior compartment of the thigh).

Table A.1: Original input time series for the Skating data sets.

A.1.1 Preprocessing

The angular displacement time series were low-pass filtered with a 2nd order Butterworth filter to remove noise from the recording process. The cutoff frequency was set to 20Hz for the ankle and knee sensor and to 10Hz for the more noisy hip angle sensor. The numerical values were converted from the measurement scale into degrees for a better interpretation. The angular velocity was derived with a 2nd order, 5 point Savitzky-Golay differentiation filter. We also estimated the angular acceleration, but the expert decided it did not offer much additional information. In addition to the regions influenced by the boundaries, another 100ms at the beginning were skipped, because of lead-in effects.

The Morlet CWT was used to analyze the time-frequency distribution of energy in the EMG signals. The relevant frequency range for EMG signals from 10Hz to 1kHz was covered by 64 equally spaced wavelet scales. The maximum width of the wavelet filter in the time domain, corresponding to 2 standard deviations, was set to 150ms. The instantaneous energy and mean frequency were extracted from the magnitude of the wavelet transform. The energy values were log transformed to deemphasize very large values and achieve a more symmetric distribution. The instantaneous mean frequency was very noisy. Especially in regions with low muscle activity, the estimates do not contain any useful information, because if a muscle is not active the measured frequency corresponds to recording noise. The frequency time series were thus discarded at this point.

A.1.2 Aspects

The 6 angular displacement and angular velocity time series were grouped into one Aspect to describe the current leg movement. We will refer to this as the *Movement* Aspect. For each EMG series a univariate Aspect was created, describing the muscle activity via the instantaneous energy. This way patterns describing the activity of the individual muscles can be found.

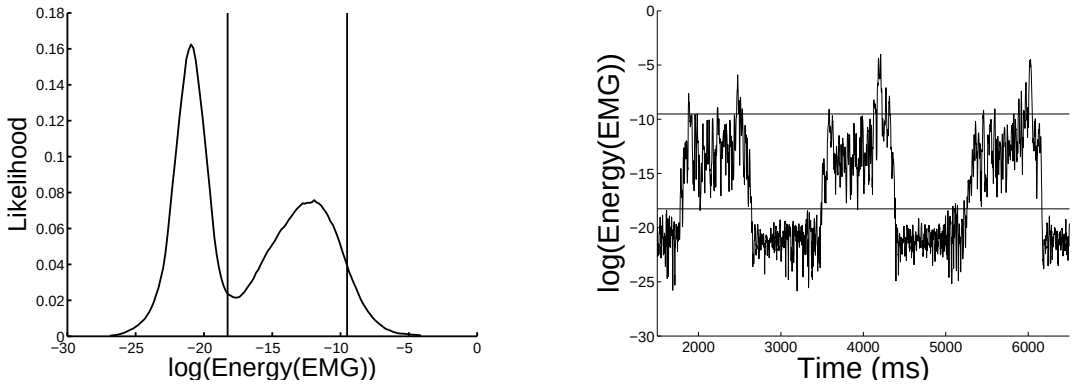
A.1.3 Tones

The Aspect for the inertia sensor was discretized into two states, namely *on* and *off*, by thresholding at 95% of the maximum.

A common method of discretizing muscle activation into states like *low* or *high* is to use thresholds at certain percentages of the maximum value. We think this methodology is highly problematic for several reasons. The maximum value and thus the thresholds are very sensitive to outliers produced by the recording process. The temporal order of values and the structure of the empirical probability density of the values, that can contain important information about possible states of the random variable, are not used by this method. Interval borders are thus possibly placed in high density regions resulting in noisy state sequences. The resulting intervals will not correspond to meaningful states of the underlying process.

The Tones were thus created using the the PERSIST (Mörchen and Ultsch, 2005b) algorithm for time series discretization. Consulting an expert we chose $k = 3$ states even though the persistence score for 4 was slightly higher. The resulting bin boundaries are shown in Figure A.1(a) as vertical lines on top of a probability density estimation plot. Note, how PERSIST places a cut to the right of the second peak. This results in a state for very high activation. The validity of this state can also be seen from Figure A.1(b), where the horizontal lines correspond to the bins selected by PERSIST. The very high values are not randomly scattered during the interval of high activation but rather concentrate toward the end of each activation phase.

The long interval of high activation is the gliding phase, where the muscle is needed to stabilize the body's center of gravity while gliding forward. The culmination of energy at the



(a) Pareto Density Estimation

(b) Time series

Figure A.1: The activation of the Gluteus muscle is discretized with bins chosen by the PERSIST algorithm.

end is explained by the push off with the foot to generate forward propulsion. The three states were labeled *low*, *high*, and *very high*, accordingly.

The Tones for the muscles Gastroc, Gluteus, and Vastus were selected for further analysis. The remaining muscles were dropped. Even though some of them showed clear states of muscle activity, they were not regarded as relevant for the major function of forward propulsion by the expert. In general, flexor muscles mainly stabilize an optimized position of the body's center of gravity with respect to the supporting area, e.g., the in-line skate. The muscles Tibial and Semiten mainly serve this purpose, therefore they constantly need to adjust and turn on and off frequently. Extensor muscles are more directly involved in forward propulsion.

The multivariate Movement Aspect was discretized using time point clustering. It can be assumed that the multivariate time series contains states corresponding to movement phases. This implies a very difficult problem for clustering algorithms, because the movement is continuous with transitions between the states. Clustering with the well known k-Means algorithm is not advisable here, because it will always find exactly as many clusters as requested, but it will not be clear whether the cluster boundaries are justified by the data. Therefore we used clustering with ESOM (Ultsch, 1999; Ultsch and Mörchen, 2005) and the accompanying U-Map (Ultsch, 1993, 2003b) visualizations.

When using ESOM, possible cluster boundaries based on distance and/or density information can be visually detected. Also, not every point needs to be assigned to a class, allowing for transition points. The goal was to find a state assignment for different movement phases, like *gliding*. The the angles and velocities were normalized to the range [0..1], i.e., the scales were transformed relative to the maximum and minimum angle or speed, respectively. A manual clustering of the ESOM was performed, Figure A.2 shows the U-Map (Ultsch, 2003b) with the 3 clusters.

Even though these clusters were found using no time information, the resulting cluster assignments are almost continuous in time, as shown in Figure A.2. The shaded bars at the bottom show the cluster membership of each time point. This validates the manual clustering of the ESOM. Using this visualization and cluster descriptions generated by the Sig* algorithm (Ultsch, 1991) an expert was able to identify and label the clusters as the following movement phases: *recovery*, *glide and push*, and *swing*.

Figure A.2: The U-Map of an ESOM trained with the Movement Aspect. The ESOM creates a topology preserving projection of the high-dimensional data space onto a 2-dimensional grid. The dark shades in the background represent large distances in the data space, bright shades indicate small distances. The data vectors of each time point are shown as a squares. Three clusters were identified manually and colored accordingly.

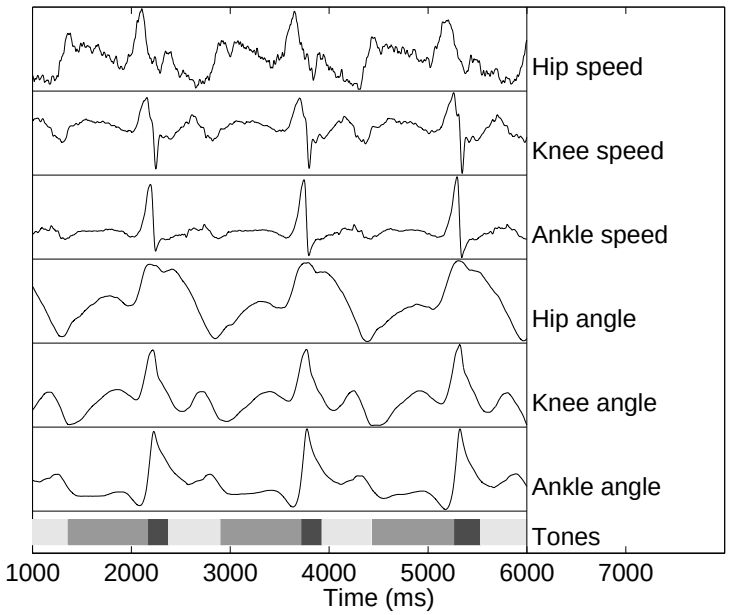


Figure A.3: Five second excerpt from Movement Aspect and the Tones obtained by ESOM clustering. The cluster memberships of the time points at the bottom forms long persisting intervals.

A.2 Video data: Recognition of Tasks from Video Recordings

This section briefly lists some filtering steps that we applied upon converting the original interval data into a set of Tones describing the actions observed in the video and the labeled segments corresponding to the different scenarios.

All predicates with the relation *supports* or *supported* were dropped, because they correspond to disjunctions of the remaining relations *contacts* and *attached*. Keeping the disjunctions would have resulted in a lot of obvious coincidences when mining Chords. We also normalized the order of the arguments in the relations by alphabetical order, e.g., converting *contacts-green-blue* to *contacts-blue-green* to obtain the exact same labels for these semantically equal situations. Finally, we replaced the term *moving* with *hand* to obtain slightly better understandable labels.

The original data contains 14 different scenarios, two version of *pick-up*, *put-down*, *stack*, *unstack*, *move*, *assemble*, and *disassemble* each. The two versions are labeled with the suffixes *-left* and right *-right* indicating reversed sides of the experimental setup. For all but the *move* scene this does not make a difference in the qualitative descriptions describing the hand and the blocks. We therefore used eight different class labels, keeping the distinction between left and right only for *move*.

The eight Tones show a promising correlation with the known scenarios in the data. Table A.2 shows the confusion matrix of Tones and scenes. The *pick-up* and *put-down* scenarios contain only tones involving the hand and the green and red block, suggesting that these experiments have exclusively been carried out using these colors. The move scenarios also seem to involve a blue block touching the red block but not the green block and the other way around for the *(un)stack* scenarios. The most complex *(dis)assemble* scenes contain other scenarios as sub steps. They involve all Tones from the *move* and *(un)stack)* scenes. In addition, the hand touching the blue block occurs exclusively during *(dis)assemble* scenes. The simple *pick-up* and *put-down* scenarios are not subtasks of the *(dis)assemble* scenes, because of interchanged block colors.

Scenes	Tones							
	contacts-blue-green	*contacts-blue-red*	*contacts-green-red*	*attached-blue-hand*	*attached-hand-red*	*attached-blue-green*	*attached-blue-red*	*attached-green-red*
pick-up	0	0	31	0	30	0	0	31
put-down	0	0	31	0	30	0	0	32
move-left	0	15	15	0	15	0	15	15
move-right	0	15	16	0	15	0	15	16
stack	60	0	30	0	30	31	0	30
unstack	60	0	31	0	30	31	0	30
assemble	63	36	0	34	34	63	34	0
disassemble	61	30	0	30	30	60	32	0

Table A.2: Occurrences of Tones in each of the scenes of the Video data.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AIC	Akaike Information Criterion
AMA	And Meets And
APCA	Adaptive Piecewise Constant Approximation
AR	Auto Regressive
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
BIC	Bayes Information Criterion
CDM	Compression-Based Dissimilarity Measure
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
DWT	Discrete Wavelet Transform
EM	Expectation Maximization
EMG	Electromyography
ESOM	Emergent Self-organizing Maps
FOL	First Order Logic
FOTL	First Temporal Order Logic
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GMM	Gaussian Mixture Model
GSP	Generalized Sequential Patterns
HMM	Hidden Markov Model
ICA	Independent Component Analysis
IFN	Info-Fuzzy Network
IQL	Interagon Query Language
KDD	Knowledge Discovery in Databases
LCSS	Longest Common Subsequence
LTL	Linear Temporal Logic
LVQ	Learning Vector Quantization
MA	Moving Average
MC	Markov Chain
MDL	Minimum Description Length
MLP	Multi Layer Perceptron
MOWCATL	Minimal Occurrences With Constraints And Time Lags
MSDD	Multi Stream Dependency Detection
MEDD	Multi Event Dependency Detection
NLG	Natural Language Generation
PAA	Piecewise Aggregate Approximation
PCA	Principal Component Analysis
PLA	Piecewise Linear Approximation

SAX	Symbolic Aggregate Approximation
SCM	Symbol Clustering Map
SDL	Shape Definition Language
SPADE	Sequential Pattern Discovery using Equivalence classes
SPAM	Sequential Pattern Mining
SOM	Self-organizing Map
STFT	Short Time Fourier Transform
SQL	Structured Query Language
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regression
TDM	Temporal Data Mining
TSDM	Time Series Data Mining
TSKM	Time Series Knowledge Mining
TSKR	Time Series Knowledge Representation
UTG	Unification-based Temporal Grammar
WTMM	Wavelet Transform Modulus Maxima

Bibliography

J. Abonyi, B. Feil, S. Németh, and A. Peter. Modified gath-geva clustering for fuzzy segmentation of multivariate time-series. *Fuzzy Sets and Systems, Data Mining Special Issue*, 149(1):39–56, 2004.

F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 12–19. ACM Press, 2004.

C. C. Aggarwal. A human-computer cooperative system for effective high dimensional clustering. In F. Provost and R. Srikant, editors, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 221–226. ACM Press, 2001.

C. C. Aggarwal. On effective classification of strings with wavelets. In D. Hand, D. Keim, and R. Ng, editors, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 163–172. ACM Press, 2002.

R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 487–499. Morgan Kaufmann, 1994.

R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 3–14. IEEE Press, 1995.

R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO'93)*, pages 69–84. Springer, 1993a.

R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of data*, pages 207–216. ACM Press, 1993b.

R. Agrawal, K. I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in times-series databases. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 490–500. Morgan Kaufmann, 1995a.

R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaot. Querying shapes of histories. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 502–514. Morgan Kaufmann, 1995b.

M. Aiello, C. Monz, L. Todoran, and M. Worring. Document understanding for a broad class of documents. *International Journal on Document Analysis and Recognition*, 5(1):1–16, 2002.

J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proceedings of the 2000 ACM Symposium on Applied Computing (SAC'00)*, pages 294–300. ACM Press, 2000.

J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26 (11):832–843, 1983.

J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 375–381. IEEE Press, 2003.

C. J. Alonso and J. J. Rodriguez. Time series classification by boosting interval based literals. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 1(11):2–11, 2000.

C. J. Alonso and J. J. Rodriguez. Boosting interval-based literals: Variable length and early classification. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, pages 145–166. World Scientific, 2004.

C. J. Alonso and J. J. Rodriguez. A graphical rule language for continuous dynamic systems. In M. Mohammadian, editor, *Computational Intelligence for Modelling, Control and Automation*, pages 482–487. IOS Press, 1999.

J. An, H. Chen, K. Furuse, N. Ohbo, and E. Keogh. Grid-based indexing for large time series databases. In J. Liu, Y. ming Cheung, and H. Yin, editors, *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'03)*, pages 614–621. Springer, 2003.

J. An, Y.-P. P. Chen, and H. Chen. DDR: An index method for large time series datasets. *Information Systems*, 30(5):333–348, 2005.

G. Andrienko, N. Andrienko, and P. Gatasky. Visual mining of spatial time series data. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pages 524–527. Springer, 2004.

M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 179–188. ACM Press, 2000.

O. Antunes. Temporal data mining: An overview. In *Workshop on Temporal Data Mining, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*. ACM Press, 2001.

A. Apostolico, M. E. Bock, S. Lonardi, and X. Xu. Efficient detection of unusual words. *Journal of Computational Biology*, 7(1-2):71–94, 2000.

A. Apostolico, F. Gong, and S. Lonardi. Verbumculus and the discovery of unusual words. *Journal of Computer Science and Technology*, 19(1):22–41, 2003.

W. G. Aref, M. G. Elfeky, and A. K. Elmagarmid. Incremental, online, and merge mining of partial periodic patterns in time-series databases. *IEEE Transactions on Knowledge and Data Engineering*, 16(3):332–342, 2004.

T. Argyros and C. Ermopoulos. Efficient subsequence matching in time series databases under time and amplitude transformations. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 481–484. IEEE Press, 2003.

J. S. Armstrong. *Principles Of Forecasting: A Handbook for Researchers and Practitioners.* Kluwer, 2001.

J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In D. Hand, D. Keim, and R. Ng, editors, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 429–435. ACM Press, 2002.

B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In L. Popa, editor, *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'02)*, pages 1–16. ACM Press, 2002.

F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 16(1-2):123–191, 2000.

K. Bach. The Top 10 Misconceptions about Implicature, for a festschrift for Larry Horn, edited by Betty Birner and Gregory Ward, published by John Benjamins, 2005. http://userwww.sfsu.edu/~kbach/TopTen.pdf.

S. Badaloni and M. Giacomin. A fuzzy extension of Allen's interval algebra. In E. Lamma and P. Mello, editors, *AI*IA99: Advances in Artificial Intelligence*, pages 155–165. Springer, 2000.

A. Bagnall, G. Janakec, and M. Zhang. Clustering time series from mixture polynomial models with discretised data. Technical Report CMP-C03-17, School of Computing Sciences, University of East Anglia, Norwich, UK, 2003.

A. J. Bagnall and G. A. Janacek. Clustering time series from ARMA models with clipped data. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 49–58. ACM Press, 2004.

A. J. Bagnall and G. J. Janacek. Clustering time series with clipped data. *Machine Learning*, 58(2-3):151–178, 2005.

J. Baixeries, G. Casas-Garriga, and J. L. Balcazar. Mining unbounded episodes from sequential data. Technical Report NC-TR-01-091, NeuroCOLT, Royal Holloway University of London, UK, 2001.

B. R. Bakshi and G. Stephanopoulos. Representation of process trends - IV. Induction of real-time patterns from operating data for diagnosis and supervisory control. *Computers & Chemical Engineering*, 18(4):303–332, 1994.

B. R. Bakshi and G. Stephanopoulos. Reasoning in time: Modeling, analysis and pattern recognition of temporal process trends. In G. Stephanopoulos and C. Han, editors, *Advances in Chemical Engineering: Paradigms of Intelligent Systems in Process Engineering*, volume 22, pages 485–547. Academic Press, 1995.

M. Bauer, U. Gather, and M. Imhoff. Analysis of high dimensional data from intensive care medicine. In R. Payne and P. Green, editors, *Proceedings in Computational Statistics*, pages 185–190. Physica, 1998.

M. Baumgarten, , A. G. Büchner, S. S. Anand, M. D. Mulvenna, and J. G. Hughes. Navigation pattern discovery from internet data. In *Workshop on Web Usage Analysis and User Profiling, 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 74–91. Springer, 1999.

N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In H. Garcia-Molina and H. V. Jagadish, editors, *Proceedings of the 1990 ACM SIGMOD International Conference on Management of data*, pages 322–331. ACM Press, 1990.

R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.

C. Berberidis, I. Vlahavas, W. G. Aref, M. Atallah, and A. K. Elmagarmid. On the discovery of weak periodicities in large time series. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'02)*, pages 51–61. Springer, 2002.

J. Beringer and E. Hüllermeyer. Online clustering of data streams. Technical Report 31, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany, 2003.

P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, USA, 2002.

D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 229–248. AAAI Press, 1996.

C. Bettini, X. Sean Wang, S. Jajodia, and J.-L. Lin. Discovering frequent event patterns with multiple granularities in time sequences. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):222–237, 1998.

M. Bicego, V. Murino, and M. Figueiredo. Similarity-based classification of sequences using Hidden Markov Models. *Pattern Recognition*, 37(12):2281–2291, 2004.

M. H. Böhlen, R. Busatto, and C. S. Jensen. Point- versus interval-based temporal data models. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, pages 192–200. IEEE Press, 1998.

B. Bollobas, G. Das, D. Gunopulos, and H. Mannila. Time-series similarity problems and well-separated geometric sets. *Nordic Journal on Computing*, 8(4):409–423, 2001.

J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In T. Terano, H. Liu, and A. L. P. Chen, editors, *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'00)*, pages 62–73. Springer, 2000.

J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: A condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery*, 7 (1):5–22, 2003.

G. Box, G. M. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting & Control*. Prentice Hall, 3rd edition, 1994.

S. Boyd. TREND: A system for generating intelligent descriptions of time-series data. In *Proceedings of the IEEE International Conference on Intelligent Processing Systems (ICIPS'98)*. IEEE Press, 1998.

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer, 2nd edition, 2002.

J. Buhler and M. Tompa. Finding motifs using random projections. In *Proceedings of the 5th Annual International Conference on Research in Computational Molecular Biology (RECOMB'01)*, pages 69–76. ACM Press, 2001.

C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):167, 1998.

A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In W. Fan, editor, *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'01)*, pages 267–273. ACM Press, 2001.

Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In G. Weikum, A. C. König, and S. Deßloch, editors, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of data*, pages 599–610. ACM Press, 2004.

T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'02)*, pages 74–85. Springer, 2002.

T. Calders and B. Goethals. Minimal k-free representations of frequent sets. In N. Lavrac, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, pages 71–82. Springer, 2003.

M. Calin and D. Galea. A fuzzy relation for comparing intervals. In B. Reusch, editor, *Proceedings of the 7th Fuzzy Days on Computational Intelligence, Theory and Applications*, pages 904–916. Springer, 2001.

G. Casas-Garriga. Discovering unbounded episodes in sequential data. In N. Lavrac, D. Gamberger, H. Blockeel, and L. Todorovski, editors, *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'03)*, pages 83–94. Springer, 2003.

G. Casas-Garriga. Summarizing sequential data with closed partial orders. In H. Kargupta, J. Srivastava, C. Kamath, , and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM'05)*, pages 380–391. SIAM, 2005.

G. Casella and E. I. George. Explaining the Gibbs sampler. *The American Statistician*, 46: 167–174, 1992.

S. Chakrabarti, S. Sarawagi, and B. Dom. Mining surprising patterns using temporal description length. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 606–617. Morgan Kaufmann, 1998.

K. Chan and A. W. Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering (ICDE'99)*, pages 126–133. IEEE Press, 1999.

K.-P. Chan, A. W.-C. Fu, and C. T. Yu. Haar wavelets for efficient similarity search of time-series: With and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, 15 (3):686–705, 2003.

J. C. Chappelier and A. Grumbach. A Kohonen map for temporal sequences. In *Proceedings of 1st International Conference on Neural Networks and Their Applications (NEURAP'95)*, pages 104–110. Domaine University Saint-Jerome, 1996.

G. Chen, X. Wu, and X. Zhu. Mining sequential patterns across data streams. Technical Report CS-05-04, University of Vermont, Burlington, VT, USA, 2005.

X. Chen and I. Petrounias. A framework for temporal data mining. In G. Quirchmayr, E. Schweighofer, and T. J. M. Bench-Capon, editors, *Proceedings 9h International Conference on Database and Expert Systems Applications (DEXA'98)*, pages 796–805. Springer, 1998.

X. Chen and I. Petrounias. Mining temporal features in association rules. In J. M. Zytkow and J. Rauch, editors, *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pages 295–300. Springer, 1999.

X. Chen, I. Petrounias, and H. Heathfield. Discovering temporal association rules in temporal databases. In M. T. Özsu, A. Dogac, and Ö. Ulusoy, editors, *Proceedings 3rd International Workshop on Issues and Applications of Database Technology (IADT'98)*, pages 312–319. Society for Design and Process Science, 1998.

H. Cheng, X. Yan, and J. Han. SeqIndex: Indexing sequences by sequential pattern analysis. In H. Kargupta, J. Srivastava, C. Kamath, , and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM'05)*, pages 84–93. SIAM, 2005.

D. W.-L. Cheung, S. D. Lee, and B. Kao. A general incremental technique for maintaining discovered association rules. In R. W. Topor and K. Tanaka, editors, *Proceedings of the 5th International Conference on Database Systems for Advanced Applications (DASFAA'97)*, pages 185–194. World Scientific, 1997.

B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 493–498. ACM Press, 2003.

M. R. Chmielewski and J. W. Grzynala-Busse. Global discretization of continuous attributes as preprocessing for machine learning. *International Journal of Approximate Reasoning*, 15: 319–331, 1996.

K. K. W. Chu and M. H. Wong. Fast time-series searching with scaling and shifting. In *Proceedings of the 18th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'99)*, pages 237–248. ACM Press, 1999.

P. R. Cohen. Fluent learning: Elucidating the structure of episodes. In F. Hoffmann, D. Hand, N. Adams, D. Fisher, and G. Guimarães, editors, *Proceedings of the 4th International Conference in Intelligent Data Analysis (IDA'01)*, pages 268–277. Springer, 2001.

P. R. Cohen and N. Adams. An algorithm for segmenting categorical time series into meaningful episodes. In F. Hoffmann, D. Hand, N. Adams, D. Fisher, and G. Guimarães, editors, *Proceedings of the 4th International Conference in Intelligent Data Analysis (IDA'01)*, pages 197–205. Springer, 2001.

P. R. Cohen, N. Adams, and D. J. Hand. Finding patterns that correspond to episodes. Technical Report UM-CS-2001-011, Experimental Knowledge Systems Laboratory, University of Massachusetts Amherst, MA, USA, 2001.

P. R. Cohen, C. Sutton, and B. Burns. Learning effects of robot actions using temporal associations. In *Proceedings of the 2nd International Conference on Development and Learning*, pages 96–101. IEEE Press, 2002.

W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. J. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 115–123. Morgan Kaufmann, 1995.

W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In S. Kambhampati and C. A. Knoblock, editors, *Workshop on Information Integration on the Web at the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 73–78, 2003.

J. Colomer, J. Melendez, and F. Gamero. Pattern recognition based on episodes and DTW. Application to diagnosis of a level control system. In *Proceedings 16th International Workshop on Qualitative Reasoning (QR'02)*, pages 37–43, 2002.

P. Cotofrei and K. Stoffel. Classification rules + time = temporal rules. In P. M. A. Sloot, C. J. K. Tan, J. Dongarra, and A. G. Hoekstra, editors, *Proceedings of the 2002 International Conference on Computational Science*, volume I, pages 572–581. Springer, 2002a.

P. Cotofrei and K. Stoffel. First-order logic based formalism for temporal data mining. In *Workshop on Foundations of Data Mining and Knowledge Discovery at the the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 101–106. IEEE Press, 2002b.

P. Cotofrei and K. Stoffel. Rule extraction from time series databases using classification trees. In *Proceedings of the 20th IASTED Conference on Applied Informatics*, pages 327–332. ACTA Press, 2002c.

P. Cotofrei and K. Stoffel. First-order logic based formalism for temporal data mining. In T. Lin, S. Ohsuga, C.-J. Liau, X. Hu, and S. Tsumoto, editors, *Foundations of Data Mining and Knowledge Discovery*, pages 193–218. Springer, 2005.

G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In H. J. Komorowski and J. M. Zytkow, editors, *Proceedings of the 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97)*, pages 88–100. Springer, 1997.

G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 16–22. AAAI Press, 1998.

D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the 5th International Conference on Intelligent Systems*, 1996.

S. Dasgupta. Experiments with random projection. In *Proceedings 16th Conference Uncertainty in Artificial Intelligence (UAI'00)*, pages 143–151. Morgan Kaufmann, 2000.

I. Daubechies. *Ten lectures on Wavelets*. SIAM, 1992.

C. S. Daw, C. E. A. Finney, and E. R. Tracy. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 74(2):916–930, 2003.

E. de Bodt, J. Rynkiewicz, and M. Cottrell. Some known facts about financial data. In M. Verleysen, editor, *Proceedings 9th European Symposium on Artificial Neural Networks (ESANN'01)*, pages 223–236. D-Facto, 2001.

A. Debregeas and G. Hebrail. Interactive interpretation of Kohonen maps applied to curves. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 179–183. AAAI Press, 1998.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, Series B(39):1–38, 1977.

K. Deng, A. Moore, and M. C. Nechyba. Learning to recognize time series: Combining ARMA models with memory-based learning. In *Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pages 246–250. IEEE Press, 1997.

A. Denton. Density-based clustering of time series subsequences. In *Workshop on Mining Temporal and Sequential Data, 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM Press, 2004.

T. G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 15–30. Springer, 2002.

P. Domingos and G. Hulten. Mining high-speed data streams. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 71–80. ACM Press, 2000.

P. Domingos and G. Hulten. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 12:945–949, 2003.

G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 43–52. ACM Press, 1999.

J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Prieditis and S. J. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ICML'95)*, pages 194–202. Morgan Kaufmann, 1995.

D. DuBois and H. Prade. Processing fuzzy temporal knowledge. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4):729–744, 1989.

A. Dugarjapov and G. Lausen. Mining sets of time series: Description of time points. In M. Schwaiger and O. Opitz, editors, *Proceedings of the 25th Annual Conference of the German Classification Society (GfKl'01)*, pages 41–49. Springer, 2002.

M. H. Dunham. *Data Mining - Introductory and Advanced Topics*. Prentice Hall, 2002.

M. G. Elfeky, W. G. Aref, and A. K. Elmagarmid. Using convolution to mine obscure periodic patterns in one pass. In E. Bertino, S. Christodoulakis, D. Plexousakis, V. C. andManolis Koubarakis, K. Böhm, and E. Ferrari, editors, *Proceedings of the 9th International Conference on Extending Database Technology (EDBT'04)*, pages 605–620. Springer, 2004.

E. Eskin. Anomaly detection over noisy data using learned probability distributions. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML'00)*, pages 255–262. Morgan Kaufmann, 2000.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithms for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231. AAAI Press, 1996.

C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In R. T. Snodgrass and M. Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of data*, pages 419–429. ACM Press, 1994.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: an overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press, 1996.

A. Fern. *Learning Models and Formulas of a Temporal Event Logic*. PhD thesis, Purdue University, West Lafayette, IN, USA, 2004.

A. Fern, R. Givan, and J. M. Siskind. Specific-to-general learning for temporal events with application to video event recognition. *Journal of Artificial Intelligence Research*, 17:379–449, 2002.

E. Fink and K. B. Pratt. Indexing of compressed time series. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, pages 43–64. World Scientific, 2004.

I. Fischer and A. Zell. String averages and self-organizing maps for strings. In H. Bothe and R. Rojas, editors, *Proceeding of the ICSC Symposia on Neural Computation (NC'2000)*, pages 208–215. ICSC Academic Press, 2000.

L. J. Fitzgibbon, D. L. Dowe, and L. Allison. Change-point estimation using new minimum message length approximations. In M. Ishizuka and A. Sattar, editors, *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence (PRICAI'02)*, pages 244–254. Springer, 2002.

J. A. Flanagan. A non-parametric approach to unsupervised learning and clustering of symbol strings and sequences. In T. Yamakawa, editor, *Proceedings of the 4th Workshop on Self-Organizing Maps (WSOM'03)*, pages 128–133, 2003.

A. Flexer and H. Bauer. Monitoring human information processing via intelligent data analysis of EEG recordings. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Proceedings of the 3rd International Conference in Intelligent Data Analysis (IDA'99)*, pages 137–148. Springer, 1999.

C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1):199–227, 1992.

A. W.-C. Fu, E. Keogh, Y. H. Lau, and C. A. Ratanamahatana. Scaling and time warping in time series querying. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P.-Å. Larson, and B. C. Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05)*, pages 649–660. Morgan Kaufmann, 2005.

T. Fu, F. Chung, V. Ng, and R. Luk. Evolutionary segmentation of financial time series into subsequences. In *Proceedings of 2001 Congress on Evolutionary Computation (CEC2001)*, pages 426–430. IEEE Press, 2001a.

T. C. Fu, F. L. Chung, V. Ng, and R. Luk. Pattern discovery from stock time series using self-organizing maps. In *Workshop on Temporal Data Mining, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 27–37. ACM Press, 2001b.

S. Gaffney and P. Smyth. Curve clustering with random effects regression mixtures. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics, 2003.

S. Gaffney and P. Smyth. Joint probabilistic curve clustering and alignment. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, number 17, pages 473–480. MIT Press, 2004.

S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 63–72. ACM Press, 1999.

X. Ge and P. Smyth. Deformable Markov model templates for time-series pattern matching. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 81–90. ACM Press, 2000.

R. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer, 1992.

P. Geurts. Pattern extraction for time series classification. In L. D. Raedt and A. Siebes, editors, *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'01)*, pages 115–127. Springer, 2001.

C. L. Giles, S. Lawrence, and A. C. Tsoi. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning*, 44(1/2):161–183, 2001.

A. Gionis and H. Mannila. Finding recurrent sources in sequences. In *Proceedings of the 7th Annual International Conference on Research in Computational Molecular Biology (RECOMB'03)*, pages 123–130. ACM Press, 2003.

A. Gionis, H. Mannila, and E. Terzi. Clustered segmentations. In *Workshop on Mining Temporal and Sequential Data, 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM Press, 2004.

D. Goldin, T. Millstein, and A. Kutlu. Flexible and efficient similarity querying for time-series data. Technical Report TR-03-4, BECAT/CSE, University of Connecticut, CN, USA, 2003.

D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*, pages 137–153. Springer, 1995.

J. Gomez, F. Gonzalez, and D. Dasgupta. An immuno-fuzzy approach to anomaly detection. In *Proceedings of The IEEE International Conference on Fuzzy Systems (FUZZIEEE)*, volume 2, pages 1219–1224. IEEE Press, 2003.

F. Gonzalez and D. Dasgupta. Neuro-immune and self-organizing map approaches to anomaly detection: A comparison. In J. Timmis and P. Bentley, editors, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS'02)*, pages 203–211. University of Kent at Canterbury, 2002.

M. Goto. A predominant-f0 estimation method for polyphonic musical audio signals. In *Proceedings of the 18th International Congress on Acoustics (ICA'04)*, pages 1085–1088. Acoustical Society of Japan, 2004.

H. Grice. *Studies in the Way of Words.* Harvard University Press, 1989.

A. Grossmann and J. Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, 15:723–736, 1984.

G. Guimarães. *Eine Methode zur Entdeckung von komplexen Mustern in Zeitreihen mit Neuronalen Netzen und deren Überführung in eine symbolische Wissensrepräsentation.* PhD thesis, Philipps-University Marburg, Germany, 1998. German.

G. Guimarães and A. Ultsch. A symbolic representation for pattern in time series using definitive clause grammars. In R. Klar and O. Opitz, editors, *Proceedings of the 20th Annual Conference of the German Classification Society (GfKl'96)*, pages 105–111. Springer, 1997.

G. Guimarães and A. Ultsch. A method for temporal knowledge conversion. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Proceedings of the 3rd International Conference in Intelligent Data Analysis (IDA'99)*, pages 369–380. Springer, 1999.

G. Guimarães, J. Peter, T. Penzel, and A. Ultsch. A method for automated temporal knowledge acquisition applied to sleep-related breathing disorders. *Artificial Intelligence in Medicine*, 23 (3):211–237, 2001.

V. Guralnik and J. Srivastava. Event detection from time series data. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 33–42. ACM Press, 1999.

D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, 1997.

R. Gwadera, M. J. Atallah, and W. Szpankowski. Markov models for identification of significant episodes. In H. Kargupta, J. Srivastava, C. Kamath, , and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM'05).* SIAM, 2005.

K. Haigh, W. Foslien, and V. Guralnik. Visual query language: Finding patterns in and relationships among time series data. In *Workshop on Mining Scientific and Engineering Datasets at the 4th SIAM International Conference on Data Mining (SDM'04)*, 2004.

J. Han and J. Pei. Pattern growth methods for sequential pattern mining: Principles and extensions. In *Workshop on Temporal Data Mining, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01).* ACM Press, 2001.

J. Han, W. Gong, and Y. Yin. Mining segment-wise periodic patterns in time-related databases. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 214–218. AAAI Press, 1998.

J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 211–218. IEEE Press, 2002.

D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

S. K. Harms and J. Deogun. Sequential association rule mining with time lags. *Journal of Intelligent Information Systems, Special issue on Data Mining*, 22(1):7–22, 2004.

S. K. Harms, J. S. Deogun, J. Saquer, and T. Tadesse. Discovering representative episodal association rules from event sequences using frequent closed episode sets and event constraints. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM'01)*, pages 603–606. IEEE Press, 2001.

S. K. Harms, J. S. Deogun, and T. Tadesse. Discovering sequential association rules with constraints and time lags in multiple sequences. In *Proceedings of the 13th International Symposium on Foundations of Intelligent Systems*, pages 432–441. Springer, 2002.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. MacMillan College Publishing Company, 1994.

G. Hebrail and B. Hugueney. Symbolic representation of long time-series. In *Workshop on Symbolic Data Analysis at the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, 2000.

M. L. Hetland. A survey of recent methods for efficient retrieval of similar time sequences. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, pages 23–42. World Scientific, 2004.

S. Hettich and S. D. Bay. The UCI KDD Archive, University of California, Irvine, CA, USA, 1999. http://kdd.ics.uci.edu.

R. J. Hilderman and H. J. Hamilton. Knowledge discovery and interestingness measures: A survey. Technical Report CS 99-04, Department of Computer Science, University of Regina, Canada, 1999.

J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmäki, and H. T. Toivonen. Time series segmentation for context recognition in mobile devices. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM'01)*, pages 203–210. IEEE Press, 2001a.

J. Himberg, J. Mäntyjärvi, and P. Korpipää. Using PCA and ICA for exploratory data analysis in situation awareness. In *Proceedings of the 2001 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI'01)*, pages 127–131. IEEE Press, 2001b.

J. Himberg, J. A. Flanagan, and J. Mäntyjärvi. Towards context awareness using symbol clustering map. In T. Yamakawa, editor, *Proceedings of the 4th Workshop on Self-Organizing Maps (WSOM'03)*, pages 249–254, 2003.

H. Hochheiser. *Interactive Graphical Querying of Time Series and Linear Sequence Data Sets*. PhD thesis, HCIL, University of Maryland, College Park, MD, USA, 2003.

F. Höppner. Discovery of temporal patterns - learning rules about the qualitative behaviour of time series. In L. D. Raedt and A. Siebes, editors, *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'01)*, pages 192–203. Springer, 2001.

F. Höppner. Discovery of core episodes from sequences - using generalization for defragmentation of rule sets. In D. Hand, R. Bolton, and N. Adams, editors, *Proceedings ESF Exploratory Workshop on Pattern Detection and Discovery in Data Mining*, pages 199–213. Springer, 2002a.

F. Höppner. Handling feature ambiguity in knowledge discovery from time series. In *Proceedings of the 7th International Conference on Discovery Science (DS'02)*, pages 398–405. Springer, 2002b.

F. Höppner. Learning dependencies in multivariate time series. In *Workshop on Knowledge Discovery in (Spatio-) Temporal Data at the 15th Eureopean Conference on Artificial Intelligence (ECAI'02)*, pages 25–31, 2002c.

F. Höppner. *Knowledge Discovery from Sequential Data.* PhD thesis, Technical University Braunschweig, Germany, 2003.

F. Höppner and F. Klawonn. Finding informative rules in interval sequences. *Intelligent Data Analysis*, 6(3):237–255, 2002.

K.-Y. Huang, C.-H. Chang, and K.-Z. Lin. Cocoa: Compressed continuity analysis for temporal databases. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pages 509–511. Springer, 2004.

G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In F. Provost and R. Srikant, editors, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 97–106. ACM Press, 2001.

J. Hunter and N. McIntosh. Knowledge-based event detection in complex time series data. In W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, and J. Wyatt, editors, *Artificial Intelligence in Medicine: Proceedings Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, pages 271–280. Springer, 1999.

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

T. Idé and K. Inoue. Knowledge discovery from heterogeneous dynamic systems using change-point correlations. In H. Kargupta, J. Srivastava, C. Kamath, , and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM'05)*. SIAM, 2005.

P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*, pages 363–372. Morgan Kaufmann, 2000.

Interagon. The Interagon Query Language - A reference guide, 2002. Interagon AS, Trondheim, Norway, http://www.interagon.com/pub/whitepapers/IQL.reference-latest.pdf.

G. J. Janacek, A. J. Bagnall, and M. Powell. A likelihood ratio distance measure for the similarity between the Fourier transform of time series. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, pages 737–743. Springer, 2005.

M. Jansen. *Noise Reduction by Wavelet Thresholding.* Springer, 2001.

H. Jeffrey. Chaos game visualization of sequences. *Computer & Graphics*, 16(1):25–33, 1992.

C. S. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. Hayes, S. J. (editors), C. Dyreson, F. Grandi, W. Käfer, N. Kline, N. Lorentzos, Y. Mitsopoulos, A. Montanari, D. Nonen, E. Peressi, B. Pernici, J. F. Roddick, N. L. Sarda, M. R. Scalas, A. Segev, R. T. Snodgrass, M. D. Soo, A. Tansel, P. Tiberio, and G. Wiederhold. A consensus glossary of temporal database concepts. *ACM SIGMOD Record*, 23(1):52–64, 1994.

L. Jiang and H. J. Hamilton. Methods for mining frequent sequential patterns. In Y. Xiang and B. Chaib-draa, editors, *Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence (AI'03)*, pages 486–491. Springer, 2003.

X. Jiang, H. Bunke, and J. Csirik. Median strings: A review. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, pages 167–192. World Scientific, 2004.

I. T. Jolliffe. *Principal Component Analysis*. Springer, 1986.

M. W. Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, University of New South Wales, Australia, 2003.

M. W. Kadous. Learning comprehensible descriptions of multivariate time series. In I. Bratko and S. Dzeroski, editors, *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pages 454–463. Morgan Kaufmann, 1999.

M. W. Kadous and C. Sammut. Constructive induction for classifying time series. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 15th European Conference on Machine Learning (ECML'04)*, pages 192–204. Springer, 2004.

M. W. Kadous and C. Sammut. Classification of multivariate time series and structured data using constructive induction. *Machine Learning*, 58(2-3):179–216, 2005.

T. Kahveci and A. K. Singh. Variable length queries for time series data. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 273–282. IEEE Press, 2001.

T. Kahveci, A. Singh, and A. Gürel. Similarity searching for multi-attribute sequences. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM'02)*, page 175. IEEE Press, 2002.

K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM'01)*, pages 273–280. IEEE Press, 2001.

P.-S. Kam and A. W.-C. Fu. Discovering temporal patterns for interval-based events. In Y. Kambayashi, M. K. Mohania, and A. M. Tjoa, editors, *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK'00)*, pages 317–326. Springer, 2000.

H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 1997.

M. B. Kennel, R. Brown, and H. D. I. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical Review A*, 45:3403, 1992.

E. Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 406–417. Morgan Kaufmann, 2002.

E. Keogh. Indexing and data mining in time series databases. personal communication, 2004.

E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In D. Hand, D. Keim, and R. Ng, editors, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 102–111. ACM Press, 2002.

E. Keogh and M. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering, and relevance feedback. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 239–241. AAAI Press, 1998.

E. Keogh and M. J. Pazzani. Scaling up dynamic time warping to massive datasets. In J. M. Zytkow and J. Rauch, editors, *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pages 1–11. Springer, 1999.

E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 24–20. AAAI Press, 1997.

E. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In W. G. Aref, editor, *Proceedings of the 2001 ACM SIGMOD International Conference on Management of data*, pages 151–162. ACM Press, 2001a.

E. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3): 263–286, 2001b.

E. Keogh, S. Lonardi, and B. Chiu. Finding surprising patterns in a time series database in linear time and space. In D. Hand, D. Keim, and R. Ng, editors, *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 550–556. ACM Press, 2002.

E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, pages 1–22. World Scientific, 2004a.

E. Keogh, S. Lonardi, and C. Ratanamahatana. Towards parameter-free data mining. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 206–215. ACM Press, 2004b.

S.-W. Kim, S. Park, and W. W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 607–614. IEEE Press, 2001.

R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 114–119. AAAI Press, 1996.

T. Kohonen. *Self-Organizing Maps*. Springer, 1995.

T. Kohonen. Self-organizing map of symbol strings. *Neurocomputing*, 21(1-3):19–30, 1999.

F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In J. Peckham, editor, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of data*, pages 289–300. ACM Press, 1997.

T. Koskela. *Neural Network Methods In Analysing And Modelling Time Varying Processes*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 2003.

J. R. Koza. Genetic programming. In J. G. Williams and A. Kent, editors, *Encyclopedia of Computer Science and Technology*, volume 39, pages 29–43. Marcel-Dekker, 1998.

M. Kryszkiewicz. Concise representation of frequent patterns based on disjunction-free generators. In N. Cercone, T. Y. Lin, and X. Wu, editors, *Proceedings of the 1st IEEE International Conference on Data Mining (ICDM'01)*, pages 305–312. IEEE Press, 2001.

M. Kryszkiewicz and M. Gajek. Concise representation of frequent patterns based on generalized disjunction-free generators. In M.-S. Cheng, P. S. Yu, and B. Liu, editors, *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)*, pages 159–171. Springer, 2002.

S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

N. Kumar, V. Lolla, E. Keogh, S. Lonardi, C. Ratanamahatana, and L. Wei. Time-series bitmaps: a practical visualization tool for working with large time series databases. In H. Kargupta, J. Srivastava, C. Kamath, , and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM'05)*, pages 531–535. SIAM, 2005.

M. Last, Y. Klein, and A. Kandel. Knowledge discovery in time series databases. *IEEE Transactions on Systems, Man, and Cybernetics*, 31(1):160–169, 2001.

G. Lausen, I. Savnik, and A. Dougarjapov. A system for mining sets of time series. In D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, editors, *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, pages 289–298. Springer, 2000.

N. Lavrac and S. Dzeroski. *Inductive Logic Programming*. Ellis Horwood, 1994.

V. Lavrenko, M. Schmill, D. Lawrie, P. Ogilvie, D. Jensen, and J. Allan. Mining of concurrent text and time series. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 37–44. ACM Press, 2000.

M. H. Law and J. T. Kwok. Rival penalized competitive learning for model-based sequence clustering. In *Proceedings of the 15th International Conference on Pattern Recognition (ICPR'00)*, volume 2, pages 195–198. IEEE Press, 2000.

S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity search for multidimensional data sequences. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, pages 599–608. IEEE Press, 2000.

N. Lesh, M. Zaki, and M. Ogihara. Mining features for sequence classification. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 342–346. AAAI Press, 1998.

C. Li and G. Biswas. Applying the Hidden Markov Model methodology for unsupervised learning of temporal data. *International Journal of Knowledge-based Intelligent Engineering Systems*, 6(3):152–160, 2002.

C. Li and G. Biswas. Temporal pattern generation using Hidden Markov Model based unsupervised classification. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Proceedings of the 3rd International Conference in Intelligent Data Analysis (IDA'99)*, pages 245–256. Springer, 1999.

C. Li, P. S. Yu, and V. Castelli. MALM: a framework for mining sequence database at multiple abstraction levels. In G. Gardarin, J. C. French, N. Pissinou, K. Makki, and L. Bouganim, editors, *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98)*, pages 267–272. ACM Press, 1998.

Q. Li, B. Moon, and I. Lopez. Skyline index for time series data. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):669–684, 2004.

Y. Li, X. S. Wang, and S. Jajodia. Discovering temporal patterns in multiple granularities. In J. F. Roddick and K. Hornsby, editors, *Proceedings of the 1st International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, pages 5–19. Springer, 2001.

J. Lin, E. Keogh, S. Lonardi, and P. Patel. Finding motifs in time series. In D. Hand, D. Keim, and R. Ng, editors, *Workshop on Temporal Data Mining, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, 2002.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 2003 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11. ACM Press, 2003a.

J. Lin, E. Keogh, and W. Truppel. Clustering of streaming time series is meaningless. In *Proceedings of the 2003 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 56–65. ACM Press, 2003b.

J. Lin, E. Keogh, S. Lonardi, J. P. Lankford, and D. M. Nystrom. Visually mining and monitoring massive time series. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 460–469. ACM Press, 2004.

J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. A MPAA-based iterative clustering algorithm augmented by nearest neighbors search for time-series data streams. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, pages 333–342. Springer, 2005.

W. Lin, M. Orgun, and G. Williams. Temporal data mining using Hidden Markov-local polynomial models. In D. Cheung, G. Williams, and Q. Li, editors, *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'01)*, pages 324–335. Springer, 2001.

T. Lindeberg. Effective scale: A natural unit for measuring scale-space lifetime. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):1068–1074, 1993.

A. Lindgren, M. T. Johnson, and R. J. Povinelli. Joint frequency domain and reconstructed phase space features for speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'04)*, pages 533–536. IEEE Press, 2004.

Z. Liu, J. X. Yu, X. Lin, H. Lu, and W. Wang. Locating motifs in time-series data. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, pages 343–353. Springer, 2005.

H. Lu, J. Han, and L. Feng. Stock movement prediction and n-dimensional inter-transaction association rules. In *Proceedings of the 1998 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12:1–12:7, 1998.

J. Ma and S. Perkins. Online novelty detection on temporal sequences. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 613–618. ACM Press, 2003.

S. Ma and J. L. Hellerstein. Mining partially periodic event patterns with unknown periods. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 205–214. IEEE Press, 2001.

J. B. MacQueen. On convergence of k-means and partitions with minimum average variance. *Annals of Mathematical Statistics*, 36(1084):1084, 1965.

O. Maimon and M. Last. *Knowledge Discovery and Data Mining, the Info-Fuzzy Network (IFN) Methodology*. Kluwer, 2000.

S. G. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

S. Manganaris. Learning to classify sensor data. Technical Report CS-95-10, Vanderbilt University, Nashville, TN, USA, 1995.

H. Mannila and C. Meek. Global partial orders from sequential data. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 161–168. ACM Press, 2000.

H. Mannila and M. Salmenkivi. Finding simple intensity descriptions from event sequence data. In F. Provost and R. Srikant, editors, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 341–346. ACM Press, 2001.

H. Mannila and J. Seppänen. Recognizing similar situations from event sequences. In R. L. Grossman, J. Han, and V. Kumar, editors, *Proceedings of the 1st SIAM International Conference on Data Mining (SDM'01)*. SIAM, 2001.

H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 146–151. AAAI Press, 1996.

H. Mannila, H. Toivonen, and I. Verkamo. Discovery of frequent episodes in event sequences. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 210–215. AAAI Press, 1995.

H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

J. Mäntyjärvi. *Sensor-based Context Recognition for Mobile Applications*. PhD thesis, Oulu University, Finland, 2003.

J. Mäntyjärvi, J. Himberg, P. Korpipää, and H. Mannila. Extracting the context of a mobile device user. In *Proceedings of the 8th IFAC IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems (IFAC HMS 2001)*, pages 450–455, 2001.

V. Megalooikonomou, G. Li, and Q. Wang. A dimensionality reduction technique for efficient similarity analysis of time series databases. In D. Grossman, L. Gravano, C. Zhai, O. Herzog, and D. A. Evans, editors, *Proceedings of the 13th International Conference on Information and Knowledge Management (CIKM'04)*, pages 160–161. ACM Press, 2004.

V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. Multiresolution symbolic representation of time series. In *Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, pages 668–679. IEEE Press, 2005.

N. Méger and C. Rigotti. Constraint-based mining of episode rules and optimal window sizes. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pages 313–324. Springer, 2004.

Merriam-Webster. Online dictionary, 2005. http://www.m-w.com.

C. S. Möller-Levet, K.-H. Cho, H. Yin, and O. Wolkenhauer. Clustering of gene expression time-series data. Technical Report 11-2003, SBI, University Rostock, Germany, 2003.

Y.-S. Moon, K.-Y. Whang, and W.-K. Loh. Efficient time-series subsequence matching using duality in constructing windows. *Information Systems*, 26(4):279–293, 2001.

C. Mooney and J. F. Roddick. Mining relationships between interacting episodes. In M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn, editors, *Proceedings of the 4th SIAM International Conference on Data Mining (SDM'04)*. SIAM, 2004.

F. Mörchen. Time series feature extraction for data mining using DWT and DFT. Technical Report 33, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany, 2003.

F. Mörchen and A. Ultsch. Discovering temporal knowledge in multivariate time series. In C. Weihs and W. Gaul, editors, *Proceedings of the 28th Annual Conference of the German Classification Society (GfKl'04)*, pages 272–279. Springer, 2005a.

F. Mörchen and A. Ultsch. Mining hierarchical temporal patterns in multivariate time series. In S. Biundo, T. W. Frühwirth, and G. Palm, editors, *Proceedings of the 27th Annual German Conference in Artificial Intelligence (KI'04)*, pages 127–140. Springer, 2004.

F. Mörchen and A. Ultsch. Optimizing time series discretization for knowledge discovery. In R. Grossman, R. Bayardo, and K. P. Bennett, editors, *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pages 660–665. ACM Press, 2005b.

F. Mörchen, A. Ultsch, M. Thies, I. Löhken, M. Nöcker, C. Stamm, N. Efthymiou, and M. Kümmerer. MusicMiner: Visualizing timbre distances of music as topographical maps. Technical Report 47, Department of Computer Sciences, Philipps-University Marburg, Germany, 2005.

F. Mörchen, A. Ultsch, and O. Hoos. Extracting interpretable muscle activation patterns with Time Series Knowledge Mining. *International Journal of Knowledge-Based & Intelligent Engineering Systems*, 2006. to appear.

K. Morik. The representation race - preprocessing for handling time phenomena. In R. L. de Mántaras and E. Plaza, editors, *Proceedings of the 11th European Conference on Machine Learning (ECML'00)*, pages 4–19. Springer, 2000.

K. Morik, M. Imboff, P. Brockhausen, T. Joachims, and U. Gather. Knowledge discovery and knowledge validation in intensive care. *Artificial Intelligence in Medicine*, 19(3):225–249, 2000.

Y. Morinaka, M. Yoshikawa, T. Amagasa, and S. Uemura. The L-index: An indexing structure for efficient subsequence matching in time sequence databases. In *Workshop on Mining Spatial and Temporal data at the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'01)*, pages 51–60, 2001.

K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, editors, *Proceedings of the 7th International Conference on Artificial Neural Networks (ICANN'97)*, pages 999–1004. Springer, 1997.

J. Mäntyjärvi, J. Himberg, P. Kangas, U. Tuomela, and P. Huuskonen. Sensor signal data set for exploring context recognition of mobile devices. In *Workshop "Benchmarks and a database for context recognition" in conjuction with the 2nd International Conference on Pervasive Computing (PERVASIVE'04)*, 2004.

G. Nagypal and B. Motik. A fuzzy model for representing uncertain, subjective and vague temporal knowledge in ontologies. In R. Meersman, Z. Tari, and D. C. Schmidt, editors, *Proceedings International Conference on Ontologies, Databases and Applications of Semantics, (ODBASE'03)*, pages 906–923. Springer, 2003.

A. Nanopoulos, R. Alcock, and Y. Manolopoulos. Feature-based classification of time-series data. In N. Mastorakis and S. D. Nikolopoulos, editors, *Information processing and technology*, pages 49–61. Nova Science, 2001.

B. Nebel and H.-J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'94)*, pages 356–361. AAAI Press, 1994.

T. Oates. PERUSE: An unsupervised algorithm for finding recurring patterns in time series. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 330–337. IEEE Press, 2002.

T. Oates. Identifying distinctive subsequences in multivariate time series by clustering. In S. Chaudhuri and D. Madigan, editors, *Fifth International Conference on Knowledge Discovery and Data Mining*, pages 322–326. ACM Press, 1999.

T. Oates, M. D. Schmill, and P. R. Cohen. Parallel and distributed search for structure in multivariate time series. Technical Report UM-CS-1996-023, Experimental Knowledge Systems Laboratory, University of Massachusetts Amherst, MA, USA, 1996.

T. Oates, M. D. Schmill, D. Jensen, and P. R. Cohen. A family of algorithms for finding temporal structure in data. In *Proceedings 6th International Workshop on Artificial Intelligence and Statistics*, 1997.

T. Oates, D. Jensen, and P. Cohen. Discovering rules for clustering and predicting asynchronous events. In *Predicting the Future: AI Approaches to Time-Series Problems, Proceedings of the 1998 Joint AAAI-ICML Workshop on Time Series Analysis*, pages 73–79. AAAI Press, 1998.

T. Oates, M. D. Schmill, and P. R. Cohen. A method for clustering the experiences of a mobile robot that accords with human judgments. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI'00)*, pages 846–851. AAAI Press / MIT Press, 2000.

T. Oates, L. Firoiu, and P. R. Cohen. Using dynamic time warping to bootstrap HMM-based clustering of time series. In R. Sun and C. L. Giles, editors, *Sequence Learning - Paradigms, Algorithms, and Applications*, pages 35–52. Springer, 2001.

H. J. Ohlbach. Relations between fuzzy time intervals. In *Proceedings 11th International Symposium on Temporal Representation and Reasoning (TIME'04)*, pages 44–51. IEEE Press, 2004.

M. Ohsaki, Y. Sato, H. Yokoi, and T. Yamaguchi. A rule discovery support system for sequential medical data – in the case study of a chronic hepatitis dataset. In *Workshop on Active Mining at the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 97–102, 2002.

A. L. Oliveira, F. B. Neto, , and S. R. Meira. Combining MLP and RBF neural networks for novelty detection in short time series. In R. Monroy, G. Arroyo-Figueroa, and L. E. Sucar, editors, *Advances in Artificial Intelligence: Proceedings 3rd Mexican International Conference on Artificial Intelligence (MICAI 2004)*, pages 844–853, Mexico City, Mexico, 2004. Springer.

J. J. Oliver, R. A. Baxter, and C. S. Wallace. Minimum message length segmentation. In X. Wu, K. Ramamohanarao, and K. B. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'98)*, pages 222–233. Springer, 1998.

P. Ormerod and C. Mounfield. Localised structures in the temporal evolution of asset prices. In *New Approaches to financial Economics, Sante Fe Conference*, 2000.

B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, pages 412–421. IEEE Press, 1998.

G. Ozsoyoglu and R. T. Snodgrass. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):513–532, 1995.

B. Padmanabhan and A. Tuzhilin. Pattern discovery in temporal databases: a temporal logic approach. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 351–354. AAAI Press, 1996.

T. Palpanas, M. Cardle, D. Gunopulos, E. Keogh, and V. B. Zordan. Indexing large human motion databases. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*, pages 780–791. Morgan Kaufmann, 2004.

A. Panuccio, M. Bicego, and V. Murino. A Hidden Markov Model-based approach to sequential data clustering. In T. Caelli, A. Amin, R. P. W. Duin, M. S. Kamel, and D. de Ridder, editors, *Proceedings Joint IAPR International Workshops Structural, Syntactic, and Statistical Pattern Recognition*, pages 734–742. Springer, 2002.

S. Park, D. Lee, and W. W. Chu. Fast retrieval of similar subsequences in long sequence databases. In *Proceedings of the 3rd IEEE Knowledge and Data Engineering Exchange Workshop*, pages 60–67. IEEE Press, 1999.

S. Park, S. Kim, and W. W. Chu. Segment-based approach for subsequence searches in sequence databases. Technical Report UCLA-CSD-TR-200022, Computer Science Department, UCLA, Los Angeles, CA, USA, 2000.

N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceeding of the 7th International Conference on Database Theory (ICDT'99)*, pages 398–416. Springer, 1999.

P. Patel, E. Keogh, J. Lin, and S. Lonardi. Mining motifs in massive time series databases. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 370–377. IEEE Press, 2002.

J. Pei, A. K. Tung, and J. Han. Fault-tolerant frequent pattern mining: Problems and challenges. In *Workshop on Research Issues in Data Mining and Knowledge Discovery, 20th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'01)*.

J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. ACM Press, 2000.

J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 215–224. IEEE Press, 2001.

J. Pei, G. Dong, W. Zou, and J. Han. On computing condensed frequent pattern bases. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 378–385. IEEE Press, 2002.

J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: The PrefixSpan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004.

C. S. Perng, H. Wang, S. R. Zhang, and D. S. Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. In *Proceedings of the 16th International Conference on Data Engineering (ICDE'00)*, pages 33–42. IEEE Press, 2000.

M. P. Perrone and S. D. Connell. K-means clustering for Hidden Markov Models. In L. R. B. Schomaker and L. G. Vuurpijl, editors, *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, pages 229–238, 2000.

S. Poliker and A. B. Geva. Non-stationary signal analysis using temporal clustering. In *The 1998 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing*, pages 304–312, Cambridge, England, 1998. IEEE Press.

I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 212–242. IEEE Press, 2002.

R. Povinelli. *Time Series Data Mining: Identifying Temporal Patterns for Characterization and Prediction of Time Series Events.* PhD thesis, Marquette University, Milwaukee, WI, USA, 1999.

R. J. Povinelli. Identifying temporal patterns for characterization and prediction of financial time series events. In J. F. Roddick and K. Hornsby, editors, *Proceedings of the 1st International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining*, pages 46–61. Springer, 2001.

R. J. Povinelli and X. Feng. A new temporal pattern identification method for characterization and prediction of complex time series events. In *IEEE Transactions on Knowledge and Data Engineering*, volume 15, pages 339–352. IEEE Press, 2003.

R. J. Povinelli, M. T. Johnson, A. C. Lindgren, and J. Ye. Time series classification using Gaussian mixture models of reconstructed phase spaces. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):779–783, 2004.

K. B. Pratt and E. Fink. Search for patterns in compressed time series. *International Journal of Image and Graphics*, 2(1):89–106, 2002.

V. Pudi and J. R. Haritsa. Generalized closed itemsets for association rule mining. In U. Dayal, K. Ramamritham, and T. M. Vijayaraman, editors, *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, pages 714–716. IEEE Press, 2003.

Y.-T. Qian, S. Jia, and W. Si. Markov model based time series similarity measuring. In *Proceedings of the 2nd IEEE International Conference on Machine Learning and Cybernetics (ICMLC'03)*, pages 278–283. IEEE Press, 2003.

Y. Qu, C. Wang, and X. S. Wang. Supporting fast search in time series for movement patterns in multiple scales. In G. Gardarin, J. C. French, N. Pissinou, K. Makki, and L. Bouganim, editors, *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98)*, pages 251–258. ACM Press, 1998.

J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufman, 1993.

R. Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 725–730. AAAI Press / MIT Press, 1996.

L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

D. Rafiei and A. O. Mendelzon. Similarity-based queries for time series data. In J. Peckham, editor, *Proceedings of the 1997 ACM SIGMOD International Conference on Management of data*, pages 13–25. ACM Press, 1997.

D. Rafiei and A. O. Mendelzon. Efficient retrieval of similar time sequences using DFT. In K. Tanaka and S. Ghandeharizadeh, editors, *Proceedings of the 5th International Conference of Foundations of Data Organization and Algorithms (FODO'98)*, pages 249–257. Springer, 1998.

C. Rainsford and J. Roddick. Adding temporal semantics to association rules. In J. M. Zytkow and J. Rauch, editors, *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pages 504–509. Springer, 1999.

C. P. Rainsford and J. F. Roddick. Visualisation of temporal interval association rules. In N. Shatin, editor, *Proceedings 2nd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'00)*, pages 91–96. Springer, 2000.

M. Ramoni, P. Sebastiani, and P. R. Cohen. Bayesian clustering by dynamics. *Machine Learning*, 47(1):91–121, 2002.

C. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi. A novel bit level time series representation with implication of similarity search and clustering. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, pages 771–777. Springer, 2005.

C. A. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *Workshop on Mining Temporal and Sequential Data at the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*. ACM Press, 2004a.

C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In M. W. Berry, U. Dayal, C. Kamath, and D. B. Skillicorn, editors, *Proceedings of the 4th SIAM International Conference on Data Mining (SDM'04)*, pages 11–22. SIAM, 2004b.

C. A. Ratanamahatana, E. Keogh, A. J. Bagnall, and S. Lonardi. A novel bit level time series representation with implications for similarity search and clustering. Technical report, University of California - Riverside, CA, USA, 2004.

T. M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. Technical Report MM-40, CIIR, University of Massachusetts at Amherst, MA, USA, 2003.

K. V. Ravi Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *ACM SIGMOD Records*, 27(2):166–176, 1998.

G. Reinert, S. Schbath, and M. S. Waterman. Probabilistic and statistical properties of words: An overview. *Journal of Computational Biology*, 7(1-2):1–46, 2000.

G. Ridgeway. Finite discrete Markov process clustering. Technical Report MSR-TR-97-24, Microsoft, Redmond, WA, USA, 1997.

J. Rissanen. *Stochastic complexity in statistical inquiry*. World Scientific, 1989.

J. F. Roddick and C. H. Mooney. Linear temporal sequences and their interpretation using midpoint relationships. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):133–135, 2005.

J. F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767, 2002.

J. J. Rodriguez and C. J. Alonso. Applying boosting to similarity literals for time series classification. In *First International Workshop on Multiple Classifier Systems*, pages 210–219. Springer, 2000.

J. J. Rodriguez, C. J. Alonso, and H. Boström. Learning first order logic time series classifiers: Rules and boosting. In D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, editors, *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, pages 299–308. Springer, 2000.

P. Ronkainen. *Attribute Similarity and Event Sequence Similarity in Data Mining.* PhD thesis, Department of Computer Science, University of Helsinki, Finland, 1998.

M. T. Rosenstein and P. R. Cohen. Concepts from time series. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, pages 739–745. AAAI Press / MIT Press, 1998.

D. Roverso. Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks. In *Proceedings of the 3rd ANS International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface Technologies, NPIC & HMIT 2000.* ANS, 2000.

P. Saetrom and M. L. Hetland. Unsupervised temporal rule mining with genetic programming and specialized hardware. In M. A. Wani, K. J. Cios, and K. Hafeez, editors, *Proceedings of the 2003 International Conference on Machine Learning and Applications (ICMLA'03)*, pages 145–151. CSREA Press, 2003.

Y. Sakurai, M. Yoshikawa, and C. Faloutsos. FTW: Fast similarity search under the time warping distance. In F. Afrati, editor, *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'05)*, pages 326–337. ACM Press, 2005.

S. Salvador and P. Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Workshop on Mining Temporal and Sequential Data, 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04).* ACM Press, 2004.

S. Salvador, P. Chan, and J. Brodie. Learning states and rules for time series anomaly detection. In V. Barr and Z. Markov, editors, *Proceedings 17th International Florida AI Research Society Conference (FLAIRS'04).* AAAI Press, 2004.

J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computation*, C-18(5):401–409, 1969.

I. Savnik, G. Lausen, H.-P. Kahle, H. Spiecker, and S. Hein. Algorithm for matching sets of time series. In D. A. Zighed, H. J. Komorowski, and J. M. Zytkow, editors, *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, pages 277–288. Springer, 2000.

R. E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1401–1406. Morgan Kaufmann, 1999.

M. Schmill, T. Oates, and P. Cohen. Learned models for continuous planning. In *Proceedings of Uncertainty 99: The 7th International Workshop on Artificial Intelligence and Statistics*, pages 278–282. Morgan Kaufmann, 1999.

A. Schmitt. *Ubiquitous computing - computing in context.* PhD thesis, Lancaster University, UK, 2002.

E. Schwalb. *Temporal Reasoning With Constraints*. PhD thesis, ICS, University of California at Irvine, CA, USA, 1998.

E. Schwalb and L. Vila. Temporal constraints: A survey. Technical report, ICS, University of California at Irvine, CA, USA, 1997.

P. Sebastiani and M. Ramoni. Clustering continuous time series. In C. E. Brodley and A. P. Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning (ICML'01)*, pages 497–504. Morgan Kaufmann, 2001.

P. Sebastiani, M. Ramoni, P. R. Cohen, J. Warwick, and J. Davis. Discovering dynamics using Bayesian clustering. In D. J. Hand, J. N. Kok, and M. R. Berthold, editors, *Proceedings of the 3rd International Conference in Intelligent Data Analysis (IDA'99)*, pages 199–210. Springer, 1999.

J. K. Seppänen and H. Mannila. Dense itemsets. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 683–688. ACM Press, 2004.

A. Sfetsos and C. Siriopoulos. Time series forecasting with a hybrid clustering scheme and pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 34(3):399–405, 2004.

C. Shahabi, X. Tian, and W. Zhao. TSA-tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries on time-series data. In *Proceedings of the 12th International Conference on Scientific and Statistical Database Management (SSDBM'00)*, pages 55–68. IEEE Press, 2000.

Y. Shahar. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90 (1-2):79–133, 1997.

H. Shatkay. Approximate queries and representations for large data sequences. Technical Report CS-95-03, Department of Computer Science, Brown University, Providence, RI, USA, 1995a.

H. Shatkay. The Fourier transform - A primer. Technical Report CS-95-37, Department of Computer Science, Brown University, Providence, RI, USA, 1995b.

H. Shatkay and S. B. Zdonik. Approximate queries and representations for large data sequences. In S. Y. W. Su, editor, *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, pages 536–545. IEEE Press, 1996.

S. W. Shaw and R. J. P. DeFigueiredo. Structural processing of waveforms as trees. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(2):328–338, 1990.

B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336. IEEE Press, 1996.

J. M. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, 15:31–90, 2001.

S. W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, 1997.

P. Smyth. Clustering sequences with Hidden Markov Models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 648. MIT Press, 1997.

P. Smyth and R. M. Goodman. Rule induction using information theory. In *Knowledge Discovery in Databases*, pages 159–176. AAAI Press / MIT Press, 1991.

C. Snoek and M. Worring. Multimedia event based video indexing using time intervals. *IEEE Transactions on Multimedia*, 7(4):638–647, 2004.

M. Spiliopoulou and L. C. Faulstich. WUM: A tool for WebUtilization analysis. In *Proceedings Extending Database Technology Workshop (WebDB '98)*, pages 184–203. Springer, 1999.

M. Spiliopoulou and J. F. Roddick. Higher order mining: Modelling and mining the results of knowledge discovery. In N. Ebecken and C. A. Brebbia, editors, *Data Mining II - Proceedings Second International Conference on Data Mining Methods and Databases*, pages 309–320. WIT Press, 2000.

R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, pages 3–17. Springer, 1996.

S. Sripada, E. Reiter, and I. Davy. SumTime-Mousam: Configurable marine weather forecast generator. *Expert Update*, 6(3):4–10, 2003a.

S. G. Sripada, E. Reiter, J. Hunter, and J. Yu. Segmenting time series for weather forecasting. In A. Macintosh, R. Ellis, and F. Coenen, editors, *Applications And Innovations in Intelligent Systems X*, pages 193–206. Springer, 2002.

S. G. Sripada, E. Reiter, and J. Hunter. Generating English summaries of time series data using the Gricean maxims. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'03)*, pages 187–196. ACM Press, 2003b.

S. G. Sripada, E. Reiter, J. Hunter, and J. Yu. Summarizing neonatal time series data. In *Proceedings of the research note sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 167–170, 2003c.

J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, 1(2): 12–23, 2000.

M. Steinbach, P.-N. Tan, and V. Kumar. Support envelopes: A technique for exploring the structure of association patterns. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, pages 296–305. ACM Press, 2004.

Z. R. Struzik. Time series rule discovery: Tough, not meaningless. In *Proceedings of the International Symposium on Methologies for Intelligent Systems*. Springer, 2003.

Z. R. Struzik and A. P. J. M. Siebes. Measuring time series' similarity through large singular features revealed with wavelet transformation. In *Proceedings of the International Workshop on Database and Expert Systems Application*, pages 162–166. IEEE Press, 1999.

X. Sun, M. E. Orlowska, and X. Zhou. Finding event-oriented patterns in long temporal sequences. In K.-Y. Whang, J. Jeon, K. Shim, and J. Srivastava, editors, *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'03)*, pages 15–26. Springer, 2003.

X. Sun, M. Orlowska, and X. Li. Extending negative event-oriented patterns in long temporal sequences. In *Workshop at the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, 2004a.

X. Sun, M. Orlowska, and X. Li. Finding negative event-oriented patterns in long temporal sequences. In H. Dai, R. Srikant, and C. Zhang, editors, *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, pages 212–221. Springer, 2004b.

X. Sun, M. E. Orlowska, and X. Li. Finding temporal features of event-oriented patterns. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, pages 778–784. Springer, 2005.

F. Takens. Detecting strange attractors in turbulence. In D. Rand and L. Young, editors, *Dynamical systems and turbulences*, pages 366–381. Springer, 1981.

Y. Tanaka and K. Uehara. Discover motifs in multi-dimensional time-series using the principal component analysis and the MDL principle. In P. Perner and A. Rosenfeld, editors, *Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'03)*, pages 252–265. Springer, 2003.

Z. Tang and P. A. Fishwick. Feed-forward neural nets as models for time series forecasting. Technical Report TR91-008, University of Florida, Gainesville, FL, USA, 1991.

P. Tino, C. Schittenkopf, and G. Dorffner. Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: Lessons learned from financial volatility trading. Technical Report 46, SFB Adaptive Information Systems and Modelling in Economics and Management Science, University Vienna, Austria, 2000.

Z. Troníček. Episode matching. In A. Amir and G. M. Landau, editors, *Proceedings of the 12th Annual Symposium of Combinatorial Pattern Matching*, pages 143–146. Springer, 2001.

A. K. H. Tung, H. Lu, J. Han, and L. Feng. Breaking the barrier of transactions: Mining inter-transaction association rules. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 297–301. ACM Press, 1999.

P. Tzvetkov, X. Yan, and J. Han. TSP: Mining Top-K closed sequential patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 347–354. IEEE Press, 2003.

A. Ultsch. Knowledge discovery, lecture notes, 2003a. German.

A. Ultsch. Maps for the visualization of high dimensional data spaces. In T. Yamakawa, editor, *Proceedings of the 4th Workshop on Self-Organizing Maps (WSOM'03)*, pages 225–230, 2003b.

A. Ultsch. Unification-based temporal grammar. Technical Report 37, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany, 2004.

A. Ultsch. Konnektionistische Modelle und ihre Integration mit wissensbasierten Systemen. Habilitationsschrift, University Dortmund, Germany, 1991.

A. Ultsch. Self-organizing neural networks for visualization and classification. In O. Opitz, B. Lausen, and R. Klar, editors, *Proceedings of the 16th Annual Conference of the German Classification Society (GfKl'92)*, pages 307–313. Springer, 1993.

A. Ultsch. Self organizing neural networks perform different from statistical k-means clustering. In H. H. Bock and W. Polasek, editors, *Proceedings of the 19th Annual Conference of the German Classification Society (GfKl'95)*, pages 1–13. Springer, 1996a.

A. Ultsch. Eine unifikationsbasierte Grammatik zur Beschreibung von komplexen Mustern in multivariaten Zeitreihen. personal notes, 1996b. German.

A. Ultsch. Data mining and knowledge discovery with emergent self-organizing feature maps for multivariate time series. In E. Oja and S. Kaski, editors, *Kohonen Maps*, pages 33–46. Elsevier, 1999.

A. Ultsch and D. Korus. Automatic acquisition of symbolic knowledge from subsymbolic neural networks. In *Proceedings 3rd European Congress on Intelligent Techniques and Soft Computing (EUFIT'95)*, pages 326–331, 1995.

A. Ultsch and F. Mörchen. ESOM-maps: Tools for clustering, visualization, and classification with Emergent SOM. Technical Report 46, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany, 2005.

J. J. van Wijk and E. R. van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'99)*, pages 4–9. IEEE Press, 1999.

K. Vasko and H. T. T. Toivonen. Estimating the number of segments in time series data using permutation tests. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 466–473. IEEE Press, 2002.

M. Vilain. A system for reasoning about time. In *Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI'82)*, pages 197–201. AAAI Press / MIT Press, 1982.

M. Vilain, H. A. Kautz, and P. G. van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In *Readings in Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kaufmann, 1989.

R. Villafane, K. A. Hua, D. Tran, and B. Maulik. Mining interval time series. In *Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, pages 318–330. Springer, 1999.

J. Vilo. *Pattern Discovery from Biosequences*. PhD thesis, Department of Computer Science, University of Helsinki, Finland, 2002.

J. Vilo. Discovering frequent patterns from strings. Technical Report C-1998-9, Department of Computer Science, University of Helsinki, Finland, 1998.

M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 673–684. IEEE Press, 2002.

M. Vlachos, J. Lin, E. Keogh, and D. Gunopulos. A wavelet-based anytime algorithm for k-means clustering of time series. In *Workshop on Clustering High Dimensionality Data and Its Applications at the 3rd SIAM International Conference on Data Mining (SDM'03)*, 2003.

M. Vlachos, D. Gunopulos, and G. Das. Indexing time-series under conditions of noise. In M. Last, A. Kandel, and H. Bunke, editors, *Data Mining In Time Series Databases*, pages 65–98. World Scientific, 2004a.

M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In G. Weikum, A. C. König, and S. Deßloch, editors, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of data*, pages 131–142. ACM Press, 2004b.

M. Vlachos, P. S. Yu, and V. Castelli. On periodicity detection and structural periodic similarity. In H. Kargupta, J. Srivastava, C. Kamath, , and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining (SDM'05)*. SIAM, 2005.

J. Wang and J. Han. BIDE: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering (ICDE'04)*, pages 79–90. IEEE Press, 2004.

W. Wang, J. Yang, and R. Muntz. TAR: Temporal association rules on evolving numerical attributes. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 283–292, Heidelberg, Germany, 2001. IEEE Press.

X. Wang, K. A. Smith, R. Hyndman, and D. Alahakoon. A scalable method for time series clustering. Technical report, Department of Econometrics and Business Statistics, Monash University, Victoria, Australia, 2004.

M. Weber, M. Alexa, and W. Müller. Visualizing time-series on spirals. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'01)*, page 7. IEEE Press, 2001.

R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 194–205. Morgan Kaufmann, 1998.

L. Wei, N. Kumar, V. Lolla, E. Keogh, S. Lonardi, and C. Ratanamahatana. Assumption-free anomaly detection in time series. In J. Frew, editor, *Proceedings of the 17th International Conference on Scientific and Statistical Database Management (SSDBM'05)*, pages 237–242, 2005.

G. M. Weiss. Mining with rarity: A unifying framework. *ACM SIGKDD Explorations Newsletter*, 6(1):7–19, 2004.

G. M. Weiss. Timeweaver: A genetic algorithm for identifying predictive patterns in sequences of events. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pages 718–725. Morgan Kaufmann, 1999.

G. M. Weiss and H. Hirsh. Learning to predict extremely rare events. In *Proceedings of the 2000 AAAI Workshop on Learning from Imbalanced Data Sets*, pages 64–68. AAAI Press, 2000.

G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 359–363. AAAI Press, 1998.

J. Wijsen. On the complexity of mining temporal trends. Technical Report 97-07, Department of Computer Science, University of British Columbia, Vancouver, Canada, 1997.

A. P. Witkin. Scale space filtering: A new approach to multi-scale description. In S. Ulmmna and R. W., editors, *Image Understanding*, pages 79–95. Ablex, 1983.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

Y.-L. Wu, D. Agrawal, and A. E. Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM'00)*, pages 488–495. ACM Press, 2000.

Y. Xiong and D. Yeung. Model-based clustering of sequential data using ARMA mixtures. In *Proceedings of the 4th ACM Postgraduate Research Day*, pages 203–210, 2003.

Y. Xiong and D. Yeung. Time series clustering with ARMA mixtures. *Pattern Recognition*, 37 (8):1675–1689, 2004.

L. Xu, A. Krzyzak, and E. Oja. Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. *IEEE Transactions on Neural Networks*, 4(4):636–648, 1993.

S. Y. and M. M.A. Knowledge-based temporal abstraction in clinical domains. *Artifical Intelligende in Medicine*, 8(3):267–98, 1996.

Y. Yamada and E. Suzuki. Decision-tree induction from time-series data based on a standard-example split test. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML'03)*, pages 840–847. Morgan Kaufmann, 2003.

X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large datasets. In D. Barbará and C. Kamath, editors, *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM'03)*, pages 166–177. SIAM, 2003.

X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In R. Grossman, R. Bayardo, and K. P. Bennett, editors, *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'05)*, pages 314–323. ACM Press, 2005.

C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In F. Provost and R. Srikant, editors, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 194–203. ACM Press, 2001a.

J. Yang, W. Wang, and P. S. Yu. Mining asynchronous periodic patterns in time series data. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 275–279. ACM Press, 2000.

J. Yang, W. Wang, and P. S. Yu. InfoMiner: Mining surprising periodic patterns. In F. Provost and R. Srikant, editors, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'01)*, pages 395–400. ACM Press, 2001b.

J. Yang, W. Wang, and P. S. Yu. InfoMiner+: Mining partial periodic patterns with gap penalties. In *Proceedings of the 2nd IEEE International Conference on Data Mining (ICDM'02)*, pages 725–728. IEEE Press, 2002.

J. Yang, W. Wang, and P. Yu. STAMP: Discovery of statistically important pattern repeats in a long sequence. In D. Barbará and C. Kamath, editors, *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM'03)*, pages 224–238. SIAM, 2003.

J. Yang, W. Wang, and P. Yu. Discovering high order periodic patterns. *Knowledge and Information Systems*, 6(3):243–268, 2004.

B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary L_p norms. In A. E. Abbadi, M. L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*, pages 385–394. Morgan Kaufmann, 2000.

B.-K. Yi, H. V. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th International Conference on Data Engineering (ICDE'98)*, pages 201–208. IEEE Press, 1998.

A. Ypma and R. P. W. Duin. Novelty detection using self-organizing maps. In K. et al., editor, *Progress in Connectionist-Based Information Systems - Proceedings of ICONIP97*, pages 1322–1325. Springer, 1997.

C.-C. Yu and Y.-L. Chen. Mining sequential patterns from multidimensional sequence data. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):136–140, 2005.

J. Yu, J. Hunter, E. Reiter, and S. Sripada. Recognising visual patterns to communicate gas turbine time-series data. In A. Macintosh, R. Ellis, and F. Coenen, editors, *Applications And Innovations in Intelligent Systems X*, pages 105–118. Springer, 2002.

M. J. Zaki. Efficient enumeration of frequent sequences. In G. Gardarin, J. C. French, N. Pissinou, K. Makki, and L. Bouganim, editors, *Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98)*, pages 68–75. ACM Press, 1998.

M. J. Zaki and C.-J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM'02)*, pages 457–473. SIAM, 2002.

M. J. Zaki and C.-J. Hsiao. Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Transaction on Knowledge and Data Engineering*, 17(4):462–478, 2005.

H. Zhang, T. B. Ho, and M. S. Lin. A non-parametric wavelet feature extractor for time-series classification. In H. Dai, R. Srikant, and C. Zhang, editors, *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04)*, pages 595–603. Springer, 2004.

Y. Zhao, C. Zhang, and S. Zhang. A recent-biased dimension reduction technique for time series data. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'05)*, pages 751–757. Springer, 2005.

S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification. In *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'02)*, volume 2, pages 1154–1159. IEEE Press, 2002.

Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In A. Y. Halevy, Z. G. Ives, and A. Doan, editors, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of data*, pages 181–192. ACM Press, 2003.

M. Zimmerman, R. J. Povinelli, M. T. Johnson, and K. Ropella. A reconstructed phase space approach for distinguishing ischemic from non-ischemic ST changes using Holter ECG data. In *Proceedings Computers in Cardiology*, 2003.

Index

Curriculum vitae

Fabian Mörchen, born 28 November 1976 in Lich, Hessen, Germany.

Education

2002 - 2006	*Ph.D. Computer Science* Philipps-University Marburg, Germany.
2001 - 2002	*M.S. Mathematics* University of Wisconsin Milwaukee, USA.
1999	*Vordiplom (B.S.) Mathematics and Computer Science* Justus-Liebig-University, Giessen, Germany.
1997 - 2001	*Studies of Mathematics and Computer Science* Justus-Liebig-University, Giessen, Germany.
1996	*Abitur (university-entrance diploma)* Wilhelm-von-Oranien Gymnasium, Dillenburg, Germany.

Employment

2002 - 2006	*Teaching assistant for Computer Science* Philipps-University Marburg, Germany.
2001 - 2002	*Teaching assistant for Mathematics* University of Wisconsin Milwaukee, USA.
2001	*Internship and freelancer* Theron Business Consulting, Cologne, Germany.
2000	*Internship* IBM Development Germany GmbH, Heidelberg, Germany.
1999 - 2001	*Programmer* Justus-Liebig-Universiy Giessen, Germany.
1997 - 1999	*Programmer* Data Becker, Germany and GData, Germany.