Universidad Autónoma de Sinaloa
Doctorate in Information Sciences

A Model for Signal Data Management and Processing

THESIS

as a requirement for the degree of
Doctor of Philosophy in Information Sciences
presented by

Oswaldo Cuen Téllez

Advisors:
Dr. Inés Fernando Vega López
Dr. Praveen Rao

Culiacán, Sinaloa, México. June 2016

Dedication

This thesis is dedicated to my parents, my wife, and my children: Mariné and Leonardo.

Acknowledgments

I would like to express my special thanks to my advisor Dr. Inés Vega. Your guidance has allowed me to grow as a professional and as a person. In particular, thank you for being a role model and for teaching me by example the values of work ethic and responsibility even when life gets rough.

I also need to thank my co-advisor Dr. Praveen Rao for allow me to work with him in the fall of 2013. Thank you for your kindness and hospitality during my doctoral stay in the University of Missouri - Kansas City.

I want to thank the Instituto Tecnólogico de Culiacán for the facilities provided during my doctoral studies. Specially, I would like to express my appreciation to M.C. Marcial Arrambí and Ing. Simón Mendoza (RIP) for their invaluable support.

I would like to acknowledge the Universidad Autónoma de Sinaloa for providing part of the funding, the physical space and the opportunity to complete my doctorate research. In particular, I want to thank Dr. Roberto Bernal for his support on arranging the funding for my doctoral trips.

Finally, my doctoral studies would have not been possible without the financial support of the Mexican Council for Science and Technology (Conacyt, scholarship No. 264305).

Contents

CONTENTS—*Continued*

List of Figures

List of Tables

Abstract

Efficient storage, processing, and analysis of signal data are significant and interrelated tasks. Scenarios such as the Internet of Things (IoT) and Big Data suggest the necessity of database systems capable of managing and analyzing very large volumes of signal data for information discovery and predictive modeling. Unfortunately, signals are not first class data types in current database technology. Consequently, database management systems lack of the proper operators for querying and processing such data type. In this dissertation, a formal data model for digital signals is proposed. This model provides a theoretical framework for the manipulation and analysis of signals in a database environment, merging the tasks of data management and data processing into a single data model. This data model consists of an abstract representation of signal data and the required set of operations for its manipulation. It is shown that this formal data model can easily be integrated with current database systems by expressing its operations using SQL statements. Moreover, using the medical field as motivation, we provide evidence that this formal data model can capture common ECG signal processing tasks needed for ECG analysis.

A new SQL clause is proposed for the implementation of sliding windows in Relational Database Management Systems (RDBMS), in the context of signal processing. This new clause can improve to a great extent the applicability of RDBMS for management and processing signal data. Furthermore, this dissertation provides a theoretical proof that shows that our proposed data model is capable of expressing any signal transformation from a finite digital signal to another finite digital signal.

Keywords: signal data management, signal data processing, signal algebra, formal data model, ECG signals

Resumen

El almacenamiento, procesamiento y análisis eficiente de datos de señales, son tareas interrelacionadas y de gran relevancia en la actualidad. El escenario que describe el *Internet de las Cosas* y la realidad del *Big Data*, sugieren la necesidad de crear sistemas de bases de datos capaces de administrar y analizar grandes volúmenes de datos de tipo señal, para el descubrimiento de información y el modelado predictivo. Desafortunadamente, las señales no se consideran datos de "primera clase" en la tecnología actual de bases de datos. En consecuencia, los gestores de bases de datos carecen de los operadores necesarios para la realización de consultas y para la ejecución de procesamiento sobre señales. En este trabajo de investigación, se propone un modelo formal para la representación de señales digitales que proporciona los fundamentos teóricos para la manipulación y análisis de señales almacenadas en bases de datos. Este modelo brinda sustento teórico tanto a la parte de administracion de datos de tipo señal, como para la parte de procesamiento de señales. La propuesta del modelo consiste en una representación abstracta de señales, así como de un conjunto de operaciones para la expresión de algoritmos de procesamiento y análisis de señales digitales. Se demuestra que este modelo puede ser fácilmente implementado en los gestores de bases de datos actuales, ya que las operaciones del modelo pueden ser traducidas, de manera directa, a consultas SQL. Además, usando el campo de la medicina como motivación, se expresan las técnicas de procesamiento de señales de Electrocardiogramas (ECG) utilizadas en el análisis de estas señales, a través el modelo propuesto.

De igual manera, se propone una nueva cláusula SQL para la implementación de ventanas deslizantes en Sistemas Gestores de Bases de Datos Relacionales (SGBDR), aplicadas al procesamiento de señales. Esta nueva cláusula permite mejorar en gran

medida la aplicabilidad de los SGBDR en el manejo y procesamiento de datos de tipo señal. Así mismo, en este trabajo de investigación se demuestra teóricamente que el modelo formal de datos propuesto es capaz de expresar cualquier transformación de una señal digital finita a otra señal digital finita.

Palabras clave: manejo de datos de señales, procesamiento de datos de señales, algebra de señales, modelo formal de datos, señales ECG

Chapter 1
Introduction

Digital signals are everywhere and play a key role in our daily life. For instance, voice, pictures, and video are all translated into digital signals in order to be stored, transmitted, recorded or displayed. A signal can be represented as a function of an independent variable, usually time, space or both. A sound signal, for instance, represents the air's vibration as a function of time at a particular point in the space [59]. Different types of signals of diverse nature are generated. Usually, signals are transformed into an electrical equivalent signal in order to be manipulated.

Electrical equivalent signals are obtained by using sensors and a sampling process. Such signals carry information and they require to be processed and analyzed in order to extract such information. The processing and analysis methods will depend on the nature of the signals. For instance, the analysis of human biomedical signals requires the involvement of an expert, usually a physician. The analysis of biomedical signals have been practiced in medical centers for several decades as a means for prevention and diagnostic of diseases. One example of such type of signals are Electrocardiograms (ECG).

ECG are records of the electrical activity (voltage) generated directly by the heart muscle cells, plotted against the time. They represent sequences of depolarization and polarization of the atria and ventricles. These records are obtained by a non-invasive procedure (Electrocardiography) consisting on positioning electrodes on the body's surface of a patient, specifically on the chest and limbs. The interface between an ECG signal source (the patient) and any acquisition device is a system of two or more

electrodes from which a differential voltage is recorded.

ECG signals contain information of can be used to diagnose Cardiovascular Diseases (CVD) [73]. This information can be extracted by analyzing ECG signals recorded simultaneously at different points of the human body and, afterwards, it can be used to support the diagnosis of CVD. CVDs are the first cause of deaths worldwide, contributing to nearly one third of global mortality. For this reason, the World Health Organization (WHO) states that the development of methods for monitoring trends of morbidity, mortality, and risk factors for developing CVD is a research priority for the prevention and control of CVD [65].

The analysis of ECG data as an aid for diagnosing CVD is a common practice in medical centers. Large volumes of ECG data are generated everyday, not only due to their usefulness but also because they are relatively easy to obtain. Keeping a data bank of the ECG data generated by hospitals from a relative large community requires important storage and data management resources. Moreover, the analysis of such data banks entails specific computation tools suitable for time evolving data such as ECG signals.

1.1 Motivation

Current vendor-supplied ECG data management systems in the market are quite limited in terms of analysis and querying. They usually support only searching and viewing the ECG data of an specific patient [69]. Moreover, large amounts of ECG data are discarded at hospitals, mostly because of the lack of effective tools to create databases where this kind of data can be not only stored, but also, efficiently queried and properly analyzed.

Global health database management systems capable of storing, querying, and analyzing very large volumes ECG data sets are needed to generate critical informa-

tion for designing public policies to prevent and control cardiovascular diseases, the number one cause of death globally [65]. This kind of systems could also serve as a decision support tool for diagnosing these diseases and for monitoring the morbidity of a population. Moreover, they can be used as a rich validation platform for new algorithms. However, before any attempt of serious implementations, such systems require new approaches for modeling signal data that can take advantage of current database technologies.

The demand for massive data analysis is not exclusive of the medical domain. The advent of mobile multi-sensor computer platforms and the ease of network connectivity have increased the collection and storage of signal data of diverse nature. Scenarios like the Internet of Things (IoT) suggests a continuous data flow from objects, devices and things as well as an unprecedented demand for data storage and analysis. Traditionally, the demand of these two tasks had been satisfied in separate ways. This is, signal data storage is usually implemented using a particular computing platform (i.e., the file system) and signal data analysis using different one (i.e., special purpose application software). This kind of approach is adequate for small size data sets, but it is not adequate for large data sets. Specially, when the amount of processed data exceeds the size of the memory available.

A formal mathematical representation of signals as a type of data is lacking. We can find in the research literature numerous algorithms for signal data processing and classification. For instance, with the purpose of automating the analysis of ECG signal data. These algorithms have been developed using techniques taken from diverse scientific disciplines such as time-series analysis, signal processing, and even image processing. Consequently, the representation of such algorithms can be as varied as the assortment of disciplines involved. Therefore, a standard mathematical expression for such algorithms can facilitate the process of translating mathematical

equations into lines of code of a programming language. Moreover, it can enable the application of machine dependent and machine independent optimization techniques in the signal data processing algorithms.

Despite the widespread generation and utilization of signal data, they are not a first class data type in Relational Database Management Systems (RDBMS). Although RDBMS are capable of storing digital signals (e.g., by using arrays), they lack of suitable operators for querying and processing this type of data. For instance, the operation of digital signal filtering, a very common signal processing operation, requires an sliding window operation between two sequences. This operation cannot be implemented by the standard operators in SQL, even though arrays are part of the SQL standard since 1999 [16]. The absence of operators for signal processing can be due to the apparent incompatible differences between scalar data (for which the relational model has been widely used as formal data model) and signal data. Consequently, a formal model for signal data is needed in order to provide this data type with a logical data structure and data operators in a database environment.

It is clear then, that in order to elevate digital signals to the category of "first class citizen" in RDBMS', it becomes necessary to create a new formal data model. This new data model should be capable of providing a suitable representation of signal data as well as being capable of expressing querying and processing algorithms. Particularly for ECG data, a formal model will simplify the implementation of ECG analysis algorithms in a RDBMS. Moreover, such formal data model can serve as a bridge between the multitude of signal analysis algorithms proposed in the literature and their expansion and implementation in large signal data repositories.

1.2 Main Contributions of this Dissertation

In this dissertation, we propose a formal model for the representation and processing of signal data in a database environment. The highlights of our proposal can be summarized as follows.

- A new model that provides a formal representation for signal data is introduced. This formal representation is suitable for its usage and implementation in current relational database technology since it is based on set theory. Moreover, we describe the alternatives for storing a signal in a RDBMS.
- In addition to the data abstraction, the proposed model defines a set of operations needed to express signal data processing and analysis.
- We present a survey of the main approaches for ECG signal data analysis and provide detailed descriptions of the processing techniques involved in such tasks. We classify signal analysis algorithms in three categories, namely, signal enhancement, feature extraction, and pattern matching.
- We provide evidence of the expressiveness of the model by using it to formally describe some of the most common techniques used in ECG signal data analysis.
- We provide a theoretical proof that the proposed model is capable of expressing any transformation from a finite digital signal into another finite digital signal.
- We describe how the proposed model can be implemented in a RDBMS by translating the set of operations into SQL statements with the aid of User Defined Functions.
- We propose a new SQL clause for the implementation of sliding windows in RDBMS, in the context of signal processing.

1.3 Organization of this Dissertation

The rest of this dissertation is organized as follows. Chapter 2 describes the fundamental techniques of signal processing. We also present a description of the processing techniques utilized in ECG analysis tasks, as well as a survey of the major algorithmic approaches for such tasks. Previous research work on algebras for digital signals and images is also presented at that chapter, as well as research opportunities. Chapter 3 explores the various forms of mathematical representations of digital signals and it relates these representations with the physical design of database systems. Moreover, we follow the process of a new database's design to describe the logical design of a basic database of ECG signals. In Chapter 4, we propose the formal model of signal data by defining the algebraic structures and the operations needed the expression of signal data analysis algorithms. In Chapter 5, we express commonly used algorithms for processing signal data using the proposed formal model, as well as a proof of the sufficiency of the model. In Chapter 6, the set of operations defined in the proposed model are translated into SQL statements. Furthermore, a new SQL clause, named `SLIDING`, is proposed for implementation of signal processing sliding windows in RDBMS. Finally, Chapter 7 summarizes the contributions of this dissertation and gives an outlook to future work.

Chapter 2
Related Work

Signal processing involves the representation and transformation of signals. Meanwhile, signal analysis is concerned with the extraction of information from signals. In this chapter, we describe the fundamental techniques commonly used in signal processing. Moreover, we describe the processing techniques utilized in ECG analysis tasks, as well as the major algorithmic approaches for such tasks. Previous research work on algebras for digital signals and images is also reviewed. At the end of this chapter, we identify some research opportunities related to signal processing, signal data analysis, and data management.

Diverse mathematical structures can be used to represent signals and their processing algorithms. Likewise, in this work, we will be using different equivalent mathematical representations. For clarity and to avoid inaccuracies in the interpretation of the mathematical objects, in Table 2.1, we show the list of mathematical symbols and notation used in this thesis.

Symbol	Description
$\mathbb{Z}$	The set of integers numbers
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}_N$	The set $\{0, 1, \ldots, N-1\}$
$\mathbf{x}$	A sequence of elements or discrete function
$x(t)$	A continuous function $\mathbf{x}$ evaluated at the real value t
$x[i]$	A discrete function $\mathbf{x}$ evaluated at the integer value i
$\mathbf{x} \star \mathbf{y}$	Discrete convolution between two sequences
$\mathbf{X}$	A discrete function represented as a set of pairs $(i, x[i])$
$\mathscr{H}$	A kernel structure

Table 2.1: List of mathematical symbols and notation used in this dissertation.

The motivation of this work is the study of the representation of ECG signals and the operations involved in processing and analysis algorithms. For this reason, we start this chapter with a description of the basics of ECG signals.

2.1 Electrocardiogram

The human body is composed of physiological processes. These processes involve complex phenomena and they are often accompanied or manifested themselves as signals that reflect the nature of such activities. The signals can be of different nature such as biochemical, electrical (voltage or current) and physical (pressure or temperature). These signals reflect properties of their associated systems, and their extraction has been found to be very helpful in identifying various pathological conditions. One of such signals are *electrocardiograms*.

Electrocardiograms (ECG) are records of an electrical signal reflecting the activity of the heart. The electric signal's (voltage) variations measured by the electrodes are caused by the action of the excitable cardiac cells as they make a heart contraction. These signals are obtained by a non-invasive procedure (Electrocardiography) con-

sisting on positioning electrodes on the body's surface of a patient, specifically on chest and limbs.

The electrodes are placed, in such a manner, that the spatiotemporal variations of the cardiac electrical field are sufficiently well-captured [11]. For an ECG recording, the difference in voltage between a pair of electrodes is referred to as a lead. Figure 2.1 illustrates the positions of the electrodes on the chest of a patient corresponding to leads $V_1 ... V_6$. These leads are also known as *precordial leads*. Additional electrodes are placed on the right arm, the left arm, the right leg and the left leg, denoted as V_{RA}, V_{LA}, V_{RL} and V_{LL}, respectively. The "central terminal" (V_{WCT}) is a reference voltage and is the average of the voltages V_{RA}, V_{LA} and V_{LL} as expressed by Equation (2.1). Precordial leads are referenced to the central terminal.

$$V_{WCT} = \frac{V_{RA} + V_{LA} + V_{LL}}{3}. \tag{2.1}$$

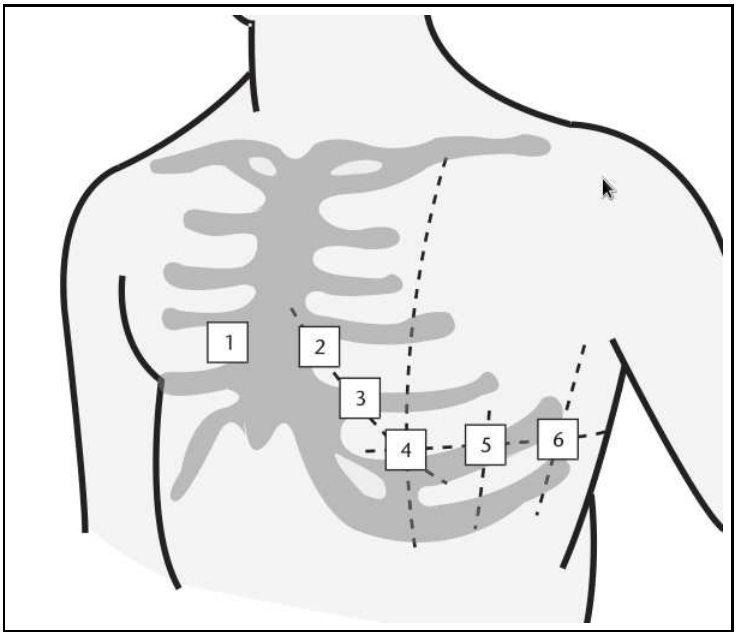

Figure 2.1: Electrodes positions on chest of a patient. Adapted from [11].

A typical ECG record consists of multiple-lead configurations which includes unipolar or bipolar leads, or both. A unipolar lead manifests the voltage variation of a single electrode and the central terminal, such as precordial leads. A bipolar lead indicates the voltage difference between two electrodes, for instance, the voltage between the left and right arm.

The standard 12-lead ECG is the most widely used ECG lead system in clinical routine [93]. The 12 leads are the six precordial leads plus the leads I, II, III, aVR, aVL, and aVF which are obtained as follows.

$$I = V_{LA} - V_{RA}, \tag{2.2}$$

$$II = V_{LL} - V_{RA}, \tag{2.3}$$

$$III = V_{LL} - V_{LA}, \tag{2.4}$$

$$aVR = V_{RA} - \frac{V_{LA} + V_{LL}}{2}, \tag{2.5}$$

$$aVL = V_{LA} - \frac{V_{RA} + V_{LL}}{2}, \tag{2.6}$$

$$aVF = V_{LL} - \frac{V_{LA} + V_{RA}}{2}, \tag{2.7}$$

These 12 signals constitute the standard 12-lead ECG. Nevertheless, the number of leads on ECG test can differ from this number, depending on the nature of the test (e.g., ambulatory monitoring), suspected cardiopathy of the patient, equipment availability, etc. [11]

2.2 Signal Processing Techniques

For several decades, there has been a significant effort to develop methods for processing and analyzing digital signals. The term digital signal processing is used here in a broad sense, comprising all algorithms that transform an input signal into another

signal. Some of the most commonly used techniques for processing digital signals are Linear Filtering, Wavelet Transforms, Polynomial Interpolation, Mathematical Morphology, and Discrete Fourier Transform. In the following subsections, we describe the fundamental digital signal processing techniques. Moreover, since this work was motivated by ECG signals, we provide a succinct description of the major approaches for the analysis of ECG signals from an algorithmic point of view. These approaches are discriminated by their purpose in the ECG analysis process.

2.2.1 Linear Filtering

Linear filters are an important class of digital signal processing systems. Besides filtering out undesired bands of frequency (noise), a linear digital filter can also be used for computing other functions such as integration, differentiation, and estimation [35]. Linear filters are modeled as Linear Time-Invariant (LTI) systems (readers who are unfamiliar with linear digital signal processing are referred to the work of Oppenheim et al. [63] for a complete description). An LTI system is completely characterized by a sequence, $\mathbf{h}$, representing the impulse response of the system.

Linear filters can be classified based on the finiteness of the sequence $\mathbf{h}$. Hence, we have two kind of filters: Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters. FIR filters are commonly used in ECG signal processing techniques such as matched and adaptive filtering. If causality (the output depends only on past and current inputs) is assumed, a FIR filter is defined as follows.

$$y[i] = \sum_{k=0}^{M-1} h[k]x[i-k] \tag{2.8}$$

This operation is called *convolution* and it is also denoted by $\mathbf{y} = \mathbf{h} \star \mathbf{x}$. Here, the sequence $\mathbf{h}$ is finite with length M. The output at any value i is obtained by

computing a weighted linear combination of the input samples $x[i], x[i-1], \ldots,$ and $x[i-M+1]$. The weights are provided by the sequence $\mathbf{h}$. This computation can be seen as an *sliding window* that considers M elements of $\mathbf{x}$ at each position. If the input sequence $\mathbf{x}$ is of length N, then the output sequence $\mathbf{y}$ will be of length $N+M-1$. Note that computing Equation (2.8) will eventually need elements not defined in $\mathbf{x}$; such as $x[-1]$. To avoid this, the length of $\mathbf{x}$ can be extended at the borders by padding zeros in order to be able to compute $\mathbf{y}$. In the case of ECG signals, M is usually much smaller than N. Therefore, in practice, we could truncate the output sequence such that its length will be N, the same as the input sequence.

In IIR filters, $\mathbf{h}$ is an infinite sequence. The convolution for an IIR filter is defined by Equation (2.9). The implementation of this expression would require an infinite number of computations for each element of the output sequence, which is unrealistic. In practice, an alternative method was derived from the analog counterpart of the IIR. In this method, besides the input sequence, the elements of the output sequence are also utilized to compute the output sequence. This recursive computation is defined by Equation (2.10).

$$y[i] = \sum_{k=0}^{\infty} h[k]x[i-k] \tag{2.9}$$

$$y[i] = \sum_{k=1}^{L} h_a[k]y[i-k] + \sum_{k=0}^{M} h_b[k]x[i-k] \tag{2.10}$$

Here, two sequences define the operation, one sequence $\mathbf{h_a}$ weights previous outputs values and a sequence $\mathbf{h_b}$ weights previous input values. Any output value is computed as a function of the previous M values of the input sequence $\mathbf{x}$ and the L previous values of the output sequence. For this reason, IIR filters are also called recursive filters. By contrast, FIR filters are usually called nonrecursive filters be-

cause only previous values of the input sequence are needed to compute the output sequence, as expressed by Equation (2.8).

Furthermore, though in theory the impulse response sequence of an IIR filter lasts forever, in practice, this sequence is finite. It decays exponentially and it could eventually sunk below the quantization step or the inherent noise of the signals [40]. Therefore, we can say that an implementation of IIR filter has an effective finite impulse response. In other words, we can implement an IIR filter with a FIR filter, if we accept the negligible error that comes from truncating the impulse response sequence.

2.2.2 Wavelet Transform

The *wavelet transform* is a linear operation that decomposes a signal into basis functions. The basis functions are obtained from a prototype wavelet, ψ, by means of dilations and contractions (scaling) as well as time shifts. If the wavelet prototype is contracted, the wavelet transform can provide information of the finer details of a signal. In the same way, if the wavelet prototype is dilated, a global view of the analyzed signal can be obtained. A discrete wavelet prototype scaled by $a > 0$, ψ_a, is defined in Equation (2.11).

$$\psi_a[i] = \frac{1}{\sqrt{a}}\psi\left[\frac{i}{a}\right] \tag{2.11}$$

The prototype wavelet can be thought as a linear filter. It can be characterized as a sequence representing the impulse response of the linear filter, as described in the previous section. Therefore, the discrete wavelet transform can be computed using linear filtering and scaling. The linear filtering is accomplished by applying the convolution operation. The scaling is obtained by subsampling [74]. In the work of

Mallat [50], the expressions for the coefficients of the discrete wavelet transform are defined. For $j \geqslant 0$, the approximation coefficients $\mathbf{a_j}$ and detail coefficients $\mathbf{d_j}$ of the discrete wavelet transform with scale j are given by Equations (2.12) and (2.13).

$$a_{j+1}[i] = \sum_{k=-\infty}^{\infty} a_j[k]h[k-2i] \tag{2.12}$$

$$d_{j+1}[i] = \sum_{k=-\infty}^{\infty} a_j[k]g[k-2i] \tag{2.13}$$

The term $\mathbf{a_0}$ represents the original signal $\mathbf{x}$. The sequence $\mathbf{h}$ is the impulse response of the linear filter representing the scaled wavelet. On the other hand, the sequence $\mathbf{g}$ represents a linear filter that computes the details of the signal that are removed by the filter $\mathbf{h}$. In a given application, the detail coefficients $\mathbf{d_j}$ are required when the lower scales of the wavelet transforms must be reconstructed from higher scales. The flow of this computation is illustrated in Figure 2.2, where Filter A and Filter B correspond to the convolution operation using the sequence $\mathbf{h}$ and sequence $\mathbf{g}$, respectively.

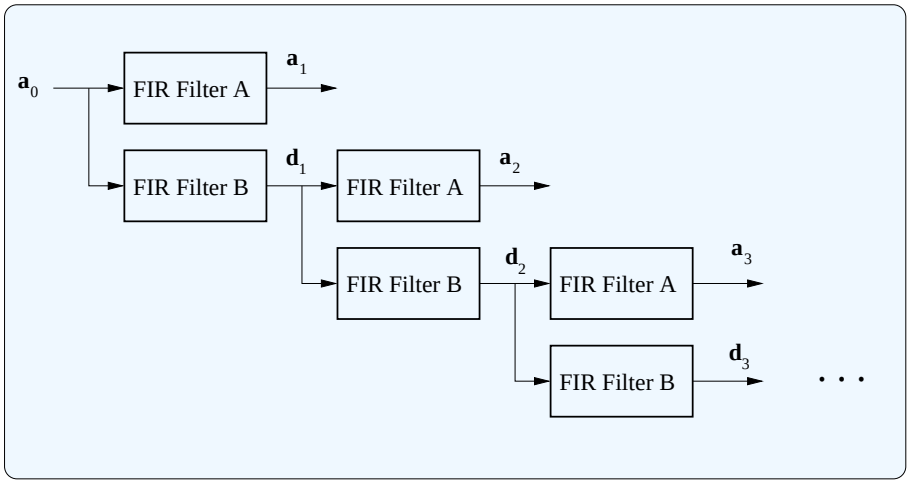

Figure 2.2: Discrete Wavelet Transform using Digital Filters.

2.2.3 Polynomial Interpolation

The task of estimating values between signal samples (interpolation) can be approached by using a parameterized polynomial model of the signals named splines. Splines can be represented by linear combinations of a special class of piecewise polynomials named B-splines. The polynomial segments are smoothly connected together at uniformly spaced samples or knots, in such a way that the interpolation function and all derivatives up to order $(n-1)$ are continuous at the knots. Splines are characterized in terms of a B-spline expansion [96] as expressed by Equation (2.14).

$$s(t) = \sum_{k=-\infty}^{\infty} c[k]\beta^m(t-k), \tag{2.14}$$

where $\mathbf{c}$ is a sequence of coefficients and β^m is a B-spline of degree m.

B-splines are symmetrical functions obtained by a $(m+1)$-fold convolution of a rectangular pulse β^0, which is defined in Equation (2.15). In Figure 2.3, the shapes of the b-splines up to degree 3 are shown.

$$\beta^m(t) = \underbrace{\beta^0 \star \beta^0 \star \beta^0 \star \ldots \star \beta^0}_{(m+1) \text{ times}}, \text{ where}$$

$$\beta^0(t) = \begin{cases} 1, & -\frac{1}{2} < t < \frac{1}{2} \\ \frac{1}{2}, & |t| = \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \tag{2.15}$$

Let us consider the spline interpolation problem where the coefficients are determined such that the function goes through the data points exactly, that is $c[k] = s[k]$. Given a sequence $\mathbf{s}$, we can determine the coefficients $\mathbf{c}$ of the B-spline model expressed by the Equation (2.14), in such a way that there is a perfect fit at the indices, as in the following expression.

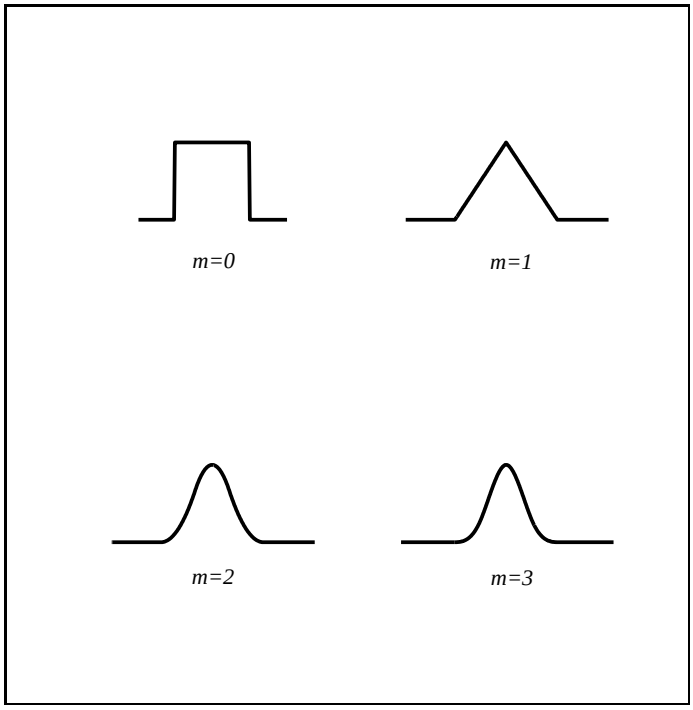

Figure 2.3: B-splines of degree 0 to 3

$$\sum_{l=-\infty}^{\infty} y[l]\beta^m(t-l)|_{t=k} = s[k] \tag{2.16}$$

In the discrete case, a B-spline kernel, $\mathbf{b}_a^m$, must be defined. This kernel is basically a sequence obtained by sampling a B-spline of degree m scaled by a, that is, $b_a^m[k] = \beta^m(t/a)|_{t=k}$. Using this kernel, we can express Equation (2.16) as $\mathbf{s} = \mathbf{b}_1^m \star \mathbf{c}$, whose solution can be found by inverse filtering as $\mathbf{c} = (\mathbf{b}_1^m)^{-1} \star \mathbf{s}$, where $(\mathbf{b}_1^m)^{-1}$ is the sequence that characterizes the inverse convolution and it can be implemented by using the convolution operation [96].

2.2.4 Mathematical Morphology

Mathematical Morphology (MM) was developed by Matheron and Serra [54, 83] in the petrography and mineralogy fields. It has been a major research topic in digital image analysis as a nonlinear method for shape-based processing. Moreover, MM operators have also been successfully applied to one dimensional signal processing tasks such as biomedical signal processing [78, 91, 98].

The basic MM operations are *erosion* ($\ominus$) and *dilation* ($\oplus$). These operations make use of a discrete structure. In the case of digital signals, the discrete structure is a sequence of elements known as the Structuring Element (SE). The SE interacts with an input sequence through the MM operations to extract relevant shapes. An specific SE has to be designed based on the particular shapes that are to be extracted for a particular application.

Let $\mathbf{x}$ be an input sequence with length N and $\mathbf{b}$ a structuring element with length M, where $N > M$. The erosion and dilation operations for signal analysis [10] are defined by Equation (2.17) and Equation (2.18), respectively.

$$(\mathbf{x} \ominus \mathbf{b})[i] = \min_{0 \leq m < M} (x[i+m] - b[m]) \text{ for } 0 \leq i \leq N-1 \tag{2.17}$$

$$(\mathbf{x} \oplus \mathbf{b})[i] = \max_{i-M+1 \leq m \leq i} (x[m] + b[i-m]) \text{ for } 0 \leq i \leq N-1 \tag{2.18}$$

Once again, some values not defined in the input signal $\mathbf{x}$ are required for computing the morphology operations. Usually, when these operators are implemented, padding techniques are used to deal with undefined values at the borders of the signal.

The basic operations, erosion and dilation, can be combined to create composed operations such as *opening* ($\circ$) and *closing* ($\bullet$). Opening performs dilation on a sequence eroded by the same SE, closing, on the other hand, performs erosion on a

previously dilated sequence.

$$\text{Opening: } x \circ b = (x \ominus b) \oplus b \tag{2.19}$$

$$\text{Closing: } x \bullet b = (x \oplus b) \ominus b \tag{2.20}$$

2.2.5 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a sequence to sequence transformation. The DFT is widely used in both, signal processing and time series data mining. Here, we consider the real version of the DFT which takes a real valued input signal and transforms it into two real valued sequences, representing the real and the imaginary parts of the DFT. The real DFT considers only operations between real numbers. The sequences involved in this transformation are the input sequence x of length N and the two output sequences, X_1 and X_0. The elements of output signal X_1 represent the cosine waves amplitudes while the elements of X_0 represent the sine wave amplitudes. The standard formulation of the real version of the DFT is given by the Equations (2.21) and (2.22) [21].

$$X_1[n] = \sum_{k=0}^{N-1} x[k] \cos(2\pi nk/N), \tag{2.21}$$

where $n \in \mathbb{Z}_{N_1}$ with

$$N_1 = \begin{cases} \frac{N}{2}, & \text{if } N \text{ is even,} \\ \frac{N-1}{2}, & \text{if } N \text{ is odd.} \end{cases}$$

$$X_0[m] = \sum_{k=0}^{N-1} x[k] \sin(2\pi mk/N), \tag{2.22}$$

where $m \in \mathbb{Z}_{N_0} - \{0\}$ with

$$N_0 = \begin{cases} \frac{N}{2} - 1, & \text{if } N \text{ is even,} \\ \frac{N-1}{2}, & \text{if } N \text{ is odd.} \end{cases}$$

Let us stress that although X_1 and X_0 represent the real and imaginary part of complex values, there are not complex operations involved in this version of the DFT.

The signal processing techniques described above present a common characteristic. All of them require an aggregation operation (sum, maximum, minimum, etc.) over a range of indices or positions. This means that all these techniques, linear and nonlinear, behave as a moving or sliding window. This remark is very important because it means that if we are able to formalize an sliding window, we will be able to create a formal model for all these processing operations. The formalization of this and other signal operations is addressed in the definition of the signal algebra in Chapter 4.

2.3 ECG Analysis Tasks

We identify three main tasks for the analysis of digital ECG signals, namely *signal enhancement*, *feature extraction*, and *pattern matching* (pattern detection and classification). The purpose of each task is described below. We also describe the major approaches that have been proposed in the literature for each task.

2.3.1 Signal Enhancement

Digital signals are susceptible to the addition of unwanted signals (noise) from different sources and characteristics. Although noise can be reduced using hardware, it is impossible to remove all of it. A corrupted signal can affect the outcome of a signal analysis process. Usually, the first step of signal analysis is the removal of distortion in order to obtain an uncorrupted signal suitable for analysis [26].

The task of noise removal from ECG signals requires an algorithm to filter out

disturbances with the least possible loss of clinical information. In other words, the morphological components of an ECG signal must get the least possible distortion from the filtering process. The ECG components are the waves, segments, and intervals illustrated in Figure 2.4. The range of frequencies of the ECG components is 0.005-150Hz. Commonly, the noise presents a wider range of frequencies compared to the ECG components. In particular, the low frequency noise (below 1Hz), can be difficult to remove from a signal because it can be in the same range of frequencies of low frequency ECG components such as the ST segment [32]. The low frequency noise in an ECG is usually caused by distance changes between electrode and the surface body due to perspiration, breathing, movements of the patient during the test, or poor electrode contact. This problem is commonly known as *baseline wander*. This kind of noise is illustrated in Figure 2.5.

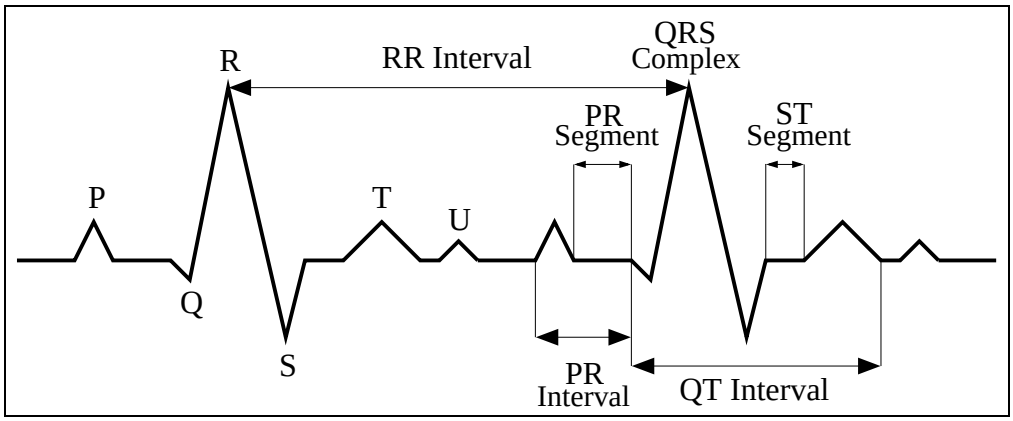

Figure 2.4: ECG waves, segments, and intervals.

Several techniques have been proposed for baseline wander removal. Some approaches make use of linear filtering to address the correction of this problem [4, 9, 13, 32, 87]. In such approaches, digital filters are designed to remove specific bands of frequency that match the frequencies of the distortion. Other methods have been proposed making use of polynomial interpolation. In these methods, it is common

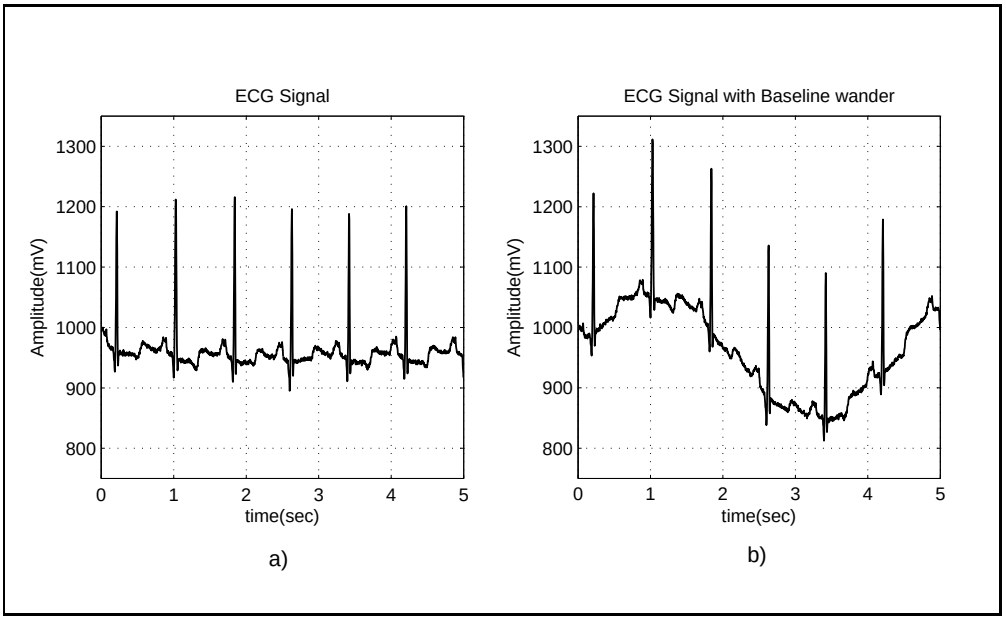

Figure 2.5: a) ECG signal b) ECG signal with baseline wander. Note that the frequency of this noise is below 1 Hz

the utilization of cubic splines [5, 57]. Cubic splines are used to estimate the noise with samples taken from PR segments, then, the estimated noise is simply subtracted from the ECG signal. Mathematical morphology approaches have been also used for baseline wander removal in [10, 61]. In the work of Sun et al. [98], a modified version of the MM operators is applied to ECG signals for baseline correction and noise suppression resulting in low distortion rate of ECG components but sacrificing noise reduction rate compared to other approaches.

Another source of noise affecting ECG signals is the one introduced by AC power-line. This kind of noise appears with a frequency of 50Hz or 60Hz. Power-Line Interference (PLI) is an electromagnetic noise added to ECG signals due to the AC power-lines. Some proposals have approached this problem by using linear filtering [4, 70]. They utilize a band-stop filter (also known as notch filter) to reject the specific frequency of the PLI. More flexible methods for PLI suppression have been proposed

by using adaptive linear filtering [52, 94]. This technique requires the modification of the filter parameters (coefficients) over time. A PLI solution involving ECG signal segmentation and linear filtering is known as the subtraction method [43, 44]. In this method, ECG signals are divided in linear and non-linear segments. Linear segments are moving-averaged by using a linear filter to remove interference. The interference is stored and further subtracted from the signal wherever non-linear segments are encountered. Some other approaches for PLI correction involves wavelets [3] and MM operators [8]. In the work of Agante and Marques de Sa [3], a wavelet approach is used for PLI correction obtaining noise reduction with only minor changes of the ECG waveforms. The work of Bhateja et al. [8] combines mathematical morphology, wavelets, and linear filters to address the problem of spikes or impulse noise and PLI. This approach starts by applying MM operators on the sequence of detail coefficients from the wavelet transform to suppress spikes, then, it applies linear filtering to the reconstructed ECG signal to remove PLI.

2.3.2 Feature Extraction

Feature extraction consists of extracting a set of characteristics from an ECG signal that captures the essence of the signal, particularly those characteristics that exhibit diagnostic properties. Most of the significant diagnostic information in an ECG signal is given by specific morphological and timing features of the ECG waves (as illustrated in Figure 2.4). The feature extraction process is accomplished by measuring time-related characteristics of the ECG components such as intervals and wave positions, and also, the amplitudes of the waves. In particular, it is of paramount importance the accurate location of the so-called fiducial points. These points determine the beginning and the end of P and T waves, the QRS complex, and the location of the R peaks [26]. The process of locating fiducial points is usually called delineation.

The QRS complex is the most distinguishable waveform of the ECG signal. The time interval of consecutive R points is used to determine the frequency of the cardiac activity. Moreover, the R point is usually taken as a reference for the delineation process. The process of locating the R point is usually known as QRS complex detection. QRS complex detection methods present a two-stage (preprocessing and decision) algorithm structure that matches most of the detectors. This structure is described by the block diagram in Figure 2.6. The preprocessing or feature enhancement stage is based on linear filtering and nonlinear transformation. The decision stage is subdivided into peak detection and decision logic. The preprocessing stage is where the original ECG signal is modified to emphasize some features associated to the QRS complex. At the same time, noise and artifacts are suppressed. On the decision stage, the individual R points are detected, usually by means of amplitude thresholds and a decision logic. R points detectors usually include adaptability to match with the changes of the signal. A postprocessing stage is often necessary for the exact determination of the temporal location of each QRS complex [39].

A broad variety of algorithms has been proposed for the QRS complex detection. Kohler et al. [39] presented an in-depth revision and comparison of QRS complex detection algorithms that use different preprocessing approaches. The linear filtering step of the QRS complex detection structure have been mostly accomplished by using convolution [60, 67], signal derivatives [31, 62], and matched filters [14, 45, 77]. All of these techniques are equivalent to linear filtering and they can be computed by using convolution [35, 39].

The wavelet transform has been widely used in QRS complex detection and delineation. Most of the wavelet approaches [1, 34, 37, 53] are based on Mallat and Hwang's work [51], where they found a relationship between modulus maxima and zero crossings of the wavelet transform of a signal and their the sharp edges.

The nonlinear filtering transformations shown in the block diagram of Figure 2.6, can be accomplished in many possible ways. Such operations are used, for example, when all elements of a signal must be transformed to positive values or when a relative suppression of small values and a slight smoothing of the peaks is required [39]. A simple nonlinear operation can be obtained by squaring each sample of the ECG signal [67], as given by Equation (2.23), where **x** is the input signal and **y** the output signal, respectively.

$$y[i] = (x[i])^2 \tag{2.23}$$

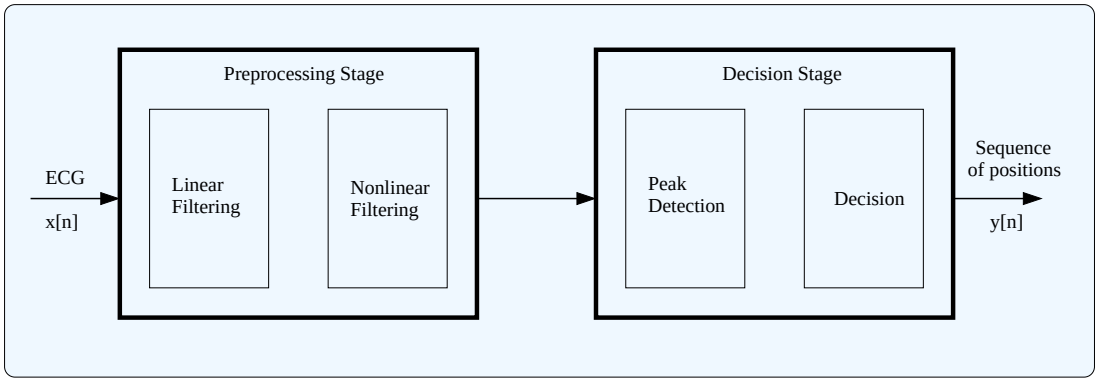

Figure 2.6: Algorithm structure of a QRS detector.Adapted from [39, 88].

Nonlinear transformations usually operate on some approximation of the derivative of the ECG signal. On digital signals, the derivative is computed using an sliding window operation. For instance, the Multiplication of Backward Difference (MOBD) algorithm [92] uses a nonlinear transformation defined as

$$y[i] = \prod_{k=0}^{M-1} |x[i-k] - x[i-k-1]| \tag{2.24}$$

The MOBD algorithm, as expressed by Equation (2.24), uses an sliding window of length 2. The product of the absolute values of the difference of the elements of

each window must be computed to obtain the transformation.

In the work of Okada [62], another example of an sliding window-based nonlinear transformation is used. A transformed signal $\mathbf{y_2}$ is obtained by squaring the difference between a linear-filtered signal $\mathbf{y_1}$ and its moving average, centered at i, using a window of length $2M + 1$, as expressed by Equation (2.25).

$$y_2[i] = \left(y_1[i] - \frac{1}{2M+1} \sum_{k=-M}^{M} y_1[i+k] \right)^2 \tag{2.25}$$

Mathematical morphology is also a window-based nonlinear transformation technique for signal analysis. This shape-based processing method has also been effectively used for QRS complex detection [95, 99, 100]. These approaches make use of different structuring elements for peaks and valleys detection. Afterwards, algorithmic approaches similar to the one shown in Figure 2.6 are usually applied.

Regarding ECG delineation algorithms, they usually define temporal search windows based on a QRS location, then, the characteristic features of each ECG wave are enhanced to facilitate their location and demarcation [53]. Some approaches for delineation [42, 56, 89] share the same algorithmic structure of QRS complex detection, as shown in Figure 2.6. However, for most of the approaches, there is not a clear rule for finding waves boundaries although the techniques utilized are basically the same techniques used in QRS detection algorithms. We can find in the literature diverse approaches to delineate ECG waves based on signal transformations such as adaptive and matched filtering [38, 86], mathematical morphology [91] and wavelet transforms [79].

2.3.3 Pattern Matching

Automated ECG pattern matching has been of great importance for real-time detection of cardiovascular diseases. The goal of ECG beat matching is to discriminate each beat and to classify it into one of a possible number of diagnostic classes [26]. Each segmented beat is defined based on the previously described delineation process. In their simplest form, ECG beat classification algorithms only use a two-class set, namely, normal and abnormal beat. More advanced algorithms are able to distinguish among a family of possible classes.

In addition to the features provided by delineation algorithms, some ECG beat classification approaches increase their accuracy by deriving a set of heuristic features that, for example, describe the area, polarity, and slopes of the waves. Furthermore, more robust approaches make use of the coefficients that result from the correlation of each beat with either a set of predefined orthonormal basis functions or a set of beat templates [88]. The classification task can be performed using techniques developed for similarity search on time series data. Similarity search requires a function to effectively measure the similarity (or dissimilarity) of a ECG beat and a template or pattern. Perhaps the most used of such functions is the L_2 Norm or Euclidean distance [25]. This kind of approach for ECG beat classification is based on the approximate matching of the beat's shape to a class template. The research literature reports high accuracy for ECG beat classification using *k-Nearest Neighbor* (k-NN) search queries [33].

The implementation Artificial Intelligence (AI) methods, especially neural networks, have become an important trend for recognition and classification of CVD [6, 41, 66]. A typical heartbeat recognition system based on neural network classifiers usually builds (trains) different models, exploiting either different classifier network

structures or different preprocessing methods of the data, and then the best one is chosen, while the rest are discarded [11]. Many other techniques have been used for ECG beat classification such as linear filtering [2], frequency domain analysis [58], and wavelet transforms [82].

2.4 Signal Algebra

The idea of defining an algebra for digital signals has been approached in different areas of signal processing. Such approaches have been made considering both, one and two dimensional (image) digital signals. An algebra can be considered as a set of objects and a collection of operations over those objects [55]. In our case, an algebra for digital signals is basically a formal representation of signals and their operations, providing a formal expression of processing and analysis algorithms.

In two dimensional signals, probably the most complete study on this matter is the work of Ritter et al. [76]. In that work, an algebra of images is proposed. The fundamental object (image) is represented using a set of pairs, where each pair consists of a coordinate and a value element. Image transformations are expressed by a set of operations of different types. Another approach is the work of D'Alotto et al. [12]. They propose an algebraic framework for image parallel processing. Here, the images are represented by bounded matrices. The algorithms are expressed using block diagrams. The processing algorithms considered are basically image to image transformations.

In the case of one dimensional digital signals, the work of Püschel and Moura [71] addresses the development of an algebraic signal processing theory, which is a generalization of linear signal processing. In that work, the signals are represented as elements of vector spaces. This theory merges the well studied areas of linear signal processing and abstract algebra (theory of groups, rings, fields, etc.). Another

interesting approach is presented in the work of Paredes et al. [68]. They focus on one dimensional signals represented as column arrays. Here again, the concepts of abstract algebra are used to investigate the properties of some linear operations and its relation to the space of one dimensional signals. They conclude that the discrete Fourier transform is an isomorphism between the space of one dimensional signals and the operations of cyclic convolution and the Hadamard product.

2.5 Time Series vs Digital Signals

In recent years, the database community has been quite active proposing alternatives for storing and querying non-conventional data such as scientific data and other forms of non-transactional data. Among these approaches, it is worth mentioning the work on time series data. Time series and digital signals are similar in the sense that they both are sequences. Therefore, we could think of taking advantage of database support for time series and use it for storing and querying signals. There is, however, one conceptual difference between these two data types. For time series, the actual time stamp for each measured value in the sequence is important. That is, the exact calendar time associated to a value is needed in the analysis [81]. For signals, this is not the case and only the order and temporal relationship between the values in a signal are of interest.

Despite the fact that temporal databases has been extensively investigated [20, 27, 28, 29, 85] there are only a few works that deal with a data model for time series data [22, 81]. Most of the proposed models are based on a predefined set of extended relational operations, their expression power with respect to time series transformation, filtering, etc., is limited [19]. In the work of Segev and Chandra [81], the data model is basically a database design for time series data. It provides the restrictions for the implementation of a database for time series analysis. The work of Faget et

al. [22] provides an extension of the relational model specifically for the management of synchronized time series data. Here, the model consists of an abstract representation of time series and a set of operations. Nevertheless, such set of operations must be defined by the users at any particular application. This means that the model does not have the sufficiency of expressing queries or processing operations over time series by itself.

2.6 Research Opportunities

Digital signal processing and analysis have made use of well studied models for the representation of signals and their algorithms. These models were thought, originally, as real-time or on-line processing models. Nevertheless, given the massive amounts of signal data produced everyday, such models require to be reconsidered. The task of processing signals is no longer an exclusive real-time process. It is impossible to approach the task of storing and processing such large signal data sets using conventional computing and signal processing solutions. Traditionally, signal data management and signal processing have been addressed separately in different computing platforms. Such approach is adequate for small size data but highly inefficient for large amounts of data. For instance, in a massive data scenario, processing tasks using the file-system and ad-hoc approaches cannot be implemented without a complex data management due to the discrepancy between data size and memory availability. Moreover, data integrity can be threaten by transferring data from one computing platform to another. On the other hand, database data management systems lack of the fundamental operations and data representation required to be a solution for the signal data management and processing needs. Consequently, it is mandatory to create the required tools to merge signal data management and processing in an efficient way.

We have identified the lack of a suitable formal model (algebra) for digital signals in a database environment. Specifically, a model where the digital signal management and signal data processing converge. A model capable of expressing both linear and nonlinear signal processing techniques using a suitable representation for signal data stored in a database environment.

In the following chapter, we explore the various forms of mathematical representations of digital signals and their relation to the physical design of database systems. Moreover, we follow the process of a new database's design to describe the logical design of a basic database of ECG signals. Moreover, in Chapter 4, we propose a signal data algebra (abstract objects and their fundamental operations) for the representation and processing of signal data in a database environment.

Chapter 3
Signal Data Management

A formal data representation is a key component for the definition of any data model. In this section, we propose a formal representation for signals. We abstract signals as sequences and express sequences as sets. We then compare this representation with the logical data abstractions used by Relational Database Management Systems (RDBMS). Throughout this thesis, we will be using the abbreviations RDBMS and DBMS interchangeably, both referring to relational DBMS.

The main motivation behind this research project is the need to define a model for digitally storing, managing and processing ECG signal data. Given this motivation, we are considering ECGs as finite digital signals. The process of analog-digital conversion is not in the scope of this work.

3.1 A Formal Representation for Finite Digital Signals

The term digital signal is here applied to those signals whose amplitude and time are both represented by discrete variables. The term discrete signal is usually associated to signals that only their time is considered discrete. For simplicity, we will consider ECG signals formally as discrete signals since this consideration is not relevant to the scope and the intended goal of this dissertation. In this work, we will use the term digital signal in a broad sense to refer to discrete-time signals.

Digital signals can be represented as sequences of numbers where, unlike the case for sets, their most important characteristic is the order. A finite sequence, $\mathbf{x}$, can be

written as

$$\mathbf{x} = (x[0], x[1], \ldots, x[N-1]) \tag{3.1}$$

or, alternatively, as $\mathbf{x} = (x[i]),$ with $x[i] \in \mathbb{R}$ and $i \in \mathbb{Z}_N = \{0, 1, \ldots, N-1\}$. Note that the index i represents the position of each element within the sequence.

We can interpret $\mathbf{x}$ as a function to represent a sequence. The domain of a function $\mathbf{x}$ is a subset of the integers and the co-domain is the set of the real numbers, as it is defined by the following expression.

$$\mathbf{x} : \mathbb{Z}_N \to \mathbb{R} \tag{3.2}$$

The term $x[i]$ is here used to refer to the function $\mathbf{x}$ evaluated at i. Note that we use brackets [ ] to indicate the discrete-time nature of the function and to differentiate it from continuous-time functions that commonly use parenthesis ( ) in the function expression. For finite length signals, prepending and appending of zeros can be used to extend its domain to the whole set of integers $\mathbb{Z}$ if needed.

In most of the signal processing literature the term $x[i]$ is used to represent an entire sequence. In this work, we refer to $x[i]$ as the "value of the i^{th} element of the sequence $\mathbf{x}$" or just the function $\mathbf{x}$ evaluated at i. Note that it is possible to expand this definition to include representations in a continuous form as $x(i\Delta_t)$, where $\Delta_t \in \mathbb{R}$ is a constant that denotes the sampling interval or period in seconds. In this case, the time index has physical units of time, but it is isomorphic to the integers [84].

An alternative way to look at a function $\mathbf{x}$ is to consider it as a special type of relation. Therefore, since a relation is simply a set of ordered pairs, we can represent a sequence or digital signal as a set. This set representation of a digital signal can be obtained using pairs of the values $x[\cdot]$ and the elements of the set of non-negative integers $\mathbb{Z}_N$, which can be regarded as an indexing set. Consequently, a sequence or digital signal can be represented by the relation $\mathbf{X}$, which it is defined as

$$\mathbf{X} = \{(i, x[i]) : i \in \mathbb{Z}_N \text{ and } x[i] \in \mathbb{R}\}. \tag{3.3}$$

This formal representation of a digital signal as a relation or set of pairs can be convenient due to the extensive usage of the relational model by Database Management Systems (DBMS). In this work, the sequences represented as sets will be denoted by uppercase bold letters.

The described representations gives us notions of the kind of data structures that can be used for storing digital signals in a DBMS. The basic elements that a data structure must consider are the index positions and their corresponding real values. There are different data structures alternatives for storing digital signals using a DBMS. These alternatives depend on the signal representation that we decide to use.

3.2 Data Structures for Storing Signals in a RDBMS

In order to effectively store digital signals using a DBMS, we have to abstract them using the data types and data structures provided by the system. We identify three alternatives for storing the elements of a sequence representing a digital signal in a conventional DBMS, namely, tuple-store, attribute-store, and *BLOBS*.

3.2.1 Tuple-Store

Using records or tuples is the straightforward method for data representation in a conventional DBMS since most major DBMS vendors utilize a record-oriented storage system [90]. In this case, digital signals can be stored using a relation named S_T with the following schema.

$$S_T\langle SId, Sequence\rangle,$$

where the SId attribute is an identification number for each signal. The attribute *Sequence* is used to represent an entire signal at each tuple. For this purpose, we use an array structure, where, in this case, each element of the array uses float data type. It is worth noting that this data type was introduced in standard SQL:1999 [16].

The tuple-store alternative has the advantage that the sequences keep their internal order because of the nature of arrays. On the other hand, this alternative requires that the whole array must be in memory in order to address particular elements of the sequence. This can be considered as a disadvantage with respect to other methods. This storage method is closely related to the sequence representation described by Equation (3.1), where each element is indexed by its position in the sequence or array.

3.2.2 Attribute-Store

An alternative for storing digital signals, the attribute-store approach, can be used in a conventional DBMS by defining a relation with two attributes as represented by the following schema.

$$S_A\langle SId, IndexId, ElemValue\rangle$$

In this case, each tuple in the relation represents just one element from a sequence. Here again, the SId attribute is an identification number for each signal. The attribute $IndexId$ holds the position value of each element in a particular sequence. The attribute $ElemValue$ corresponds to the values of each of the element of the sequence.

Each row of relation S_A holds an element of a sequence. Unlike arrays, where the order is implicit, here we need to sort the elements of a particular signal to arrange them in their original order. The tuples produced by a query can be ordered on

attribute $IndexId$ using the SQL `ORDER BY` clause.

Since each element of the sequence is indeed a tuple from a relation, this method is the actual database implementation of the abstract representation of a sequence described in Equation (3.3), which uses pairs $(i, x[i])$ to represent elements in a sequence.

3.2.3 BLOBS

There is a third alternative that considers a type-less container, a special kind of attribute in a database relation. This is actually a similar storing method to the tuple-store. The difference resides in that here we are using a generic container for storing a sequence instead of using the array data type. Data stored in such containers are usually known as Binary Large OBjects (BLOBs). Here, we use a relation named as S_B to store a digital signal with two attributes as represented by the following schema.

$$S_B\langle SId, Sequence\rangle$$

A drawback of using BLOBs as a method for storing data is the lack of operations defined by the DBMS over such attributes. Moreover, it is well known that storing large-volume data (such an ECG) in the form of BLOBs has a negative impact on the performance of the database [80]. It also has significant storing overhead compared to a filesystem storage [48].

3.3 Logical Modeling of ECG Signal Data in a RDBMS

A typical design process of a new database involves the transformation of a high-level logical design of data into relational database schemas. The final step of this process is the implementation of such schemas in a DBMS. In this section, we follow the

process of a new database's design to describe the logical design of a basic database of ECG signals and to see how the relational database schemas can be obtained from the logical design. We use the Entity/Relationship (E/R) Model to logically model the signal data. Afterwards, we transform E/R diagrams into database schemas.

3.3.1 The Entity/Relationship Model of an ECG Signal Database

ECGs are represented by signal data. The particular characteristics of ECG signals were described in Chapter 2. Here, we use the E/R model to logically describe an ECG signal database. The E/R model is used to represent data through diagrams using three type of elements: entities, attributes and relationships. An entity represents a collection of real life objects and it is represented as a rectangle in the E/R diagram. The attributes are the properties associated to an entity set and they are represented as ovals. The relationships are connections between entities and they are represented as diamonds in a E/R diagram.

The logical model obtained from the E/R analysis will lead us to use specific data structures to store ECG signals. We can classify the data structures alternatives for signals in two categories. BLOBs and Tuple-Store in one category and Attribute-Store in another one. This classification is based on the nature of the objects used to store the signals. In the case of BLOBs and Tuple-Store, the signals are logically stored in a single object, whether a BLOB or an array. The design of a ECG signal database that leads us to use this type of storage can be described by a E/R diagram as shown in the Figure 3.1. The diagram represents ECG signal data following the approach of single logical object. The E/R diagram has two entities and one relationship. One entity is called *ECG*. This entity represents every electrocardiogram test performed, let say, in a network of hospitals across the country. Each ECG test will have the following attributes: `IdECG`, `SamplingFreq`, `LengthTime` and `Mode`. In the attribute `IdECG`

represents a identification for each electrocardiogram. The attribute `SamplingFreq` gives information about the sampling frequency of a particular ECG. `LengthTime` gives the length in time units of the ECG. Finally, the attribute `Mode` is used to describe the type test that was performed such as an exercise ECG, holter or other types of ECG tests. On the other hand, the entity named *Signals* represents the signals associated with an electrocardiogram. The attributes of the *Signals* entity are: `LeadName`, and `Sequence`. The attribute `LeadName` gives information about the lead name of the signal. The attribute `Sequence` represents the actual signal object. The relationship `HAS` connects each ECG with a set of signals and it is a one to many relationship. That is, for each ECG there could be N signals.

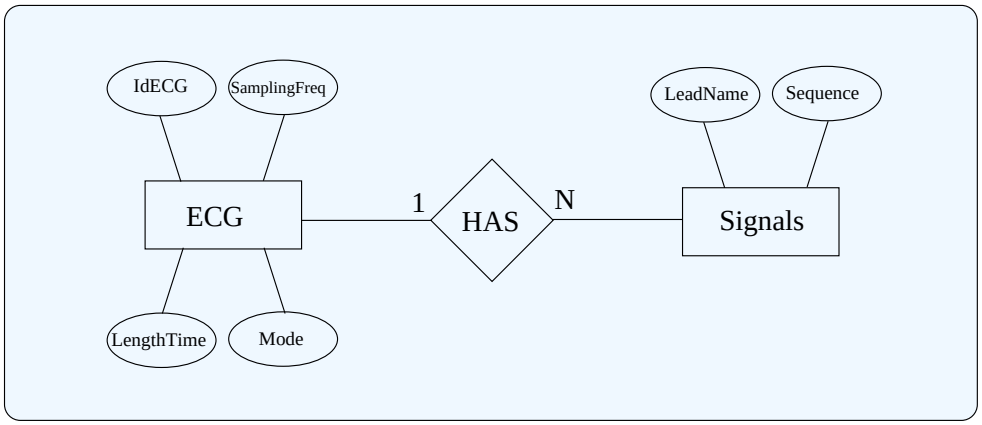

Figure 3.1: E/R Diagram for the BLOBS and Tuple-Store approach.

An alternative logical design of an ECG signal database that requires the use of the Attribute-Store approach is illustrated by the E/R diagram of the Figure 3.2. Here, the entities *ECG* and *Signals*, and the relationship `HAS` remain the same, except that the attribute `Sequence` is not present in this diagram. The entity set *Values* is now included in this approach. The attributes of *Values* are `TimeIndex` and `Value` and they represent the position and the value of each element of the sequence or signal,

respectively. The relationship `CONTAINS` establishes the link between each signal and its values and it is a one to many relationship.

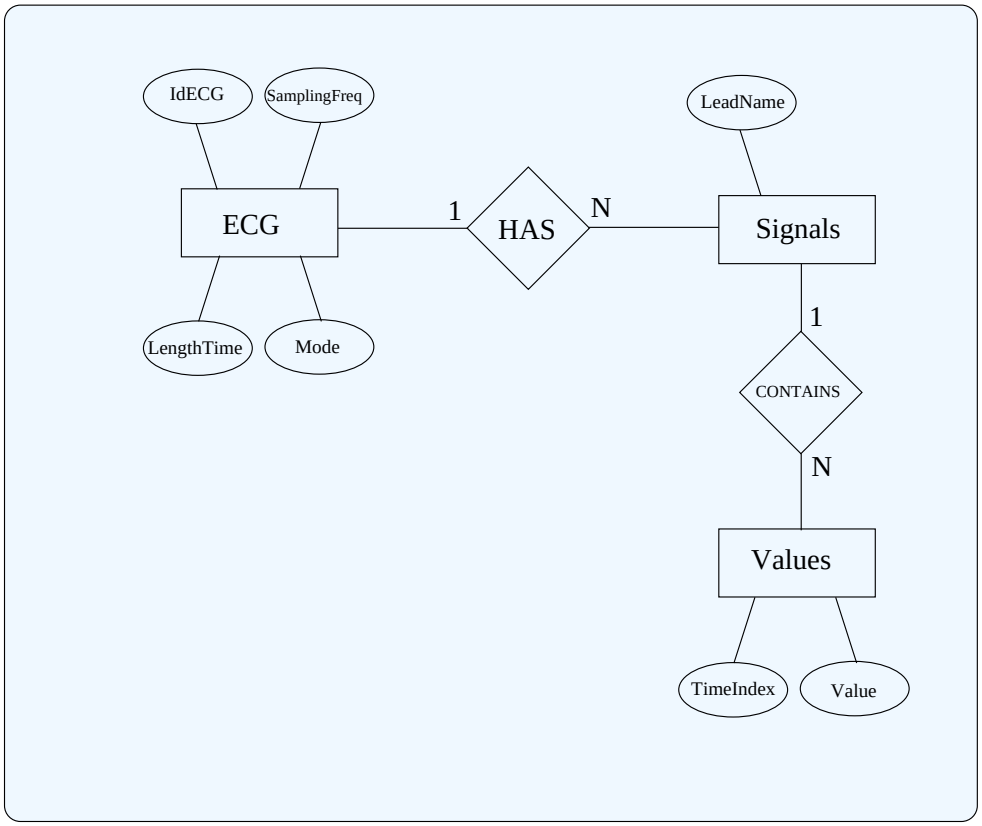

Figure 3.2: E/R Diagram for the Attribute-Store approach.

3.3.2 Database Relational Schemas of an ECG Signal Database

In order to translate the previous E/R diagrams into database relational schemas, we do the following steps:

- Turn each entity set into a relation with the same set of attributes, and
- Replace a relationship by a relation whose attributes are the keys for the connected entity sets.

Let us consider the two entity sets *ECG* and *Signals* from the E/R diagram of Figure 3.1. The attributes for the *ECG* entity set are `IdECG`, `SamplingFreq`, `LengthTime`, and `Mode`. As a result, a typical instance of the relation ECG is shown in Figure 3.3. Next, consider the entity set *Signals* from Figure 3.1. There are two attributes, `LeadName`, and `Sequence`. Thus, we would expect the corresponding *Signals* relation to have schema $Signals\langle LeadName, Sequence\rangle$ and a typical instance is shown in Figure 3.4. Similarly, the relationship HAS shown in Figure 3.1 can be transformed into a relation with the attributes `IdECG`, `LeadName`, and `Sequence`. The relation *HAS* is shown in Figure 3.5.

IdECG	SamplingFreq	LengthTime	Mode
C786	100	24	Holter
A578	200	0.5	Exercise
J456	250	1.5	Standard

Figure 3.3: A sample instance of the ECG relation.

LeadName	Sequence
V1	[-0.116996,-0.118197,-0.114594 . . . -0.067752,-0.068954,-0.059345]
V6	[0.091990,0.107604,0.141234 . . . -0.137415,-0.138398,-0.154230]
V3	[0.515968,0.458317,0.393459 . . . 0.087186, 0.055958, 0.017524]
V1	[-0.160235,-0.157833,-0.179452 . . . -0.074959,-0.077361,-0.074959]
aVR	[-0.091774,-0.094176,-0.102584 . . . 0.361030,0.287765,0.225309]
aVL	[0.021127,0.042746,0.052355 . . . -0.002895,-0.022112,-0.028117]

Figure 3.4: A sample instance of the Signals relation.

In the case of the E/R diagram shown in Figure 3.2, the previous schemas also apply. The only difference is that here, the attribute `Sequence` of the entity set *Signals* is not necessary. Now, the relationship *CONTAINS* is the link between the entity set *Signals* and *Values*. The relationship *CONTAINS* is a many to many relationship

and it can be transformed into a relation. An instance of the relation *CONTAINS* is shown in the Figure 3.6.

IdECG	LeadName	Sequence
J456	V1	[-0.116996,-0.118197,-0.114594 ...-0.067752,-0.068954,-0.059345]
J456	V6	[0.091990,0.107604,0.141234 ...-0.137415,-0.138398,-0.154230]
A578	V3	[0.515968,0.458317,0.393459 ...0.087186, 0.055958, 0.017524]
C786	V1	[-0.160235,-0.157833,-0.179452 ...-0.074959,-0.077361,-0.074959]
A578	aVR	[-0.091774,-0.094176,-0.102584 ...0.361030,0.287765,0.225309]
A578	aVL	[0.021127,0.042746,0.052355 ...-0.002895,-0.022112,-0.028117]

Figure 3.5: A sample instance of the HAS relation.

LeadName	TimeIndex	Value
V1	0	-0.116996
V1	1	-0.118197
V1	2	-0.114594
⋮	⋮	⋮
V1	8639997	-0.067752
V1	8639998	-0.068954
V1	8639999	-0.059345
aVR	0	-0.091774
aVR	1	-0.094176
aVR	2	-0.102584
⋮	⋮	⋮
aVR	1349997	0.361030
aVR	1349998	0.287765
aVR	1349999	0.225309

Figure 3.6: A sample instance of the CONTAINS relation.

Chapter 4
An Algebra for Signal Data Processing

In this chapter, we describe our proposed formal data model for signals. We start by defining a formal representation of signals. After this, we define the set of operations necessary to express signal transformation algorithms. Our proposed model is based on the work of Ritter et al. [76] where an algebra for processing images is proposed. Since one of our goals is to provide a formal model for signal data management, we limit our attention to finite real-valued signals which can be stored and manipulated by a computer. Therefore, we start with the definition of an abstract object to represent a signal.

4.1 Signals as Relations

As previously described, a sequence is a suitable representation of a digital signal. It can be mathematically defined as a set of pairs, particularly a function. Consequently, we use a set pairs to be the formal representation of a signal in this proposal. The reason for this is twofold. First, set theory is a mature and well developed area of mathematics. Second, a set representation fits well with the database theory (the relational model) which is based on the set theory. This choice allows us to use an abstract representation that is suitable for both, signal data management and signal data processing.

Definition 1 (Sequence or Signal)**.** *A real-valued sequence of length N can be represented as a function $\mathbf{X} : \mathbb{Z}_N \to \mathbb{R}$. The set of all functions $\mathbb{Z}_N \to \mathbb{R}$ is denoted by $\mathbb{R}^{\mathbb{Z}_N}$. The set $\mathbb{R}$ is the set of all possible values for each element of the sequence*

and $\mathbb{Z}_N$ is the set of all possible index positions in the sequence and will be named the indexing set. Therefore, a sequence $\mathbf{X}$ is defined by the following set of pairs.

$$\mathbf{X} = \{(i, x[i]) : x[i] \in \mathbb{R},\ i \in \mathbb{Z}_N\} \tag{4.1}$$

Each element in the sequence is represented by a pair $(i, x[i])$. The first element of the pair, i, is an index that specifies the position of the element in the sequence. The second element, $x[i]$, represents the value associated with position i. In other words, $x[i]$ is a sample value of a digital signal at i. The cardinality of this set corresponds to the length of the sequence. Let us define the length of a sequence using the cardinality of a set denoted by $|\cdot|$, thus, the length of the sequence $\mathbf{X}$ can be expressed as $|\mathbf{X}| = N$.

A particular kind of sequence where all the N sample values $(x[i])$ are the same is called a constant sequence.

Definition 2 (Constant Sequence)**.** *A sequence with the same value at every index position will be named a* constant sequence*. Let us consider a sequence $\mathbf{Z} \in \mathbb{R}^{\mathbb{Z}_N}$ defined as,*

$$\mathbf{Z} = \{(i, K) : K \in \mathbb{R},\ i \in \mathbb{Z}_N\}, \tag{4.2}$$

which is a constant sequence with a value K at every index position.

Some useful constant sequences are those consisting of only ones and only zeros. We will denote these sequences as $\mathbf{1} = \{(i, 1) : i \in \mathbb{Z}_N\}$ and $\mathbf{0} = \{(i, 0) : i \in \mathbb{Z}_N\}$, respectively. Without loss of generality, it is assumed that when $\mathbf{1}$ and $\mathbf{0}$ are used in a sequence expression, they have the necessary length to be consistent with the sequences and operations involved in the expression.

4.2 Sequence Operations

Now, we proceed to define operations for processing sequences. Since the elements of the sequences are real values, the binary operations between sequences can be compared to the binary operations defined on the set of real numbers.

Definition 3 (Binary Operations Between Sequences)**.** *Let* $\mathbf{X}$, $\mathbf{Y} \in \mathbb{R}^{\mathbb{Z}_N}$ *be two sequences of the same length. Let* $\odot$ *be a binary operation on* $\mathbb{R}$*, then, the result of a binary operation* $\odot$ *between two sequences is the sequence* $\mathbf{Z} = \mathbf{X} \odot \mathbf{Y}$ *defined in the following expression.*

$$\mathbf{Z} = \mathbf{X} \odot \mathbf{Y} = \{(i, c[i]) : c[i] = x[i] \odot y[i],\ i \in \mathbb{Z}_N\} \tag{4.3}$$

We define three binary operations between sequences. All of them are commutative and associative for sequences. Let $\mathbf{X}$, $\mathbf{Y} \in \mathbb{R}^{\mathbb{Z}_N}$ be two sequences, the binary operations of addition (+), multiplication ($\times$) and maximum (max) are defined as follows.

$$\mathbf{X} + \mathbf{Y} = \{(i, c[i]) : c[i] = x[i] + y[i],\ i \in \mathbb{Z}_N\} \tag{4.4}$$

$$\mathbf{X} \times \mathbf{Y} = \{(i, c[i]) : c[i] = x[i] \times y[i],\ i \in \mathbb{Z}_N\} \tag{4.5}$$

$$^{1}\max(\mathbf{X}, \mathbf{Y}) = \{(i, c[i]) : c[i] = \max(x[i], y[i]),\ i \in \mathbb{Z}_N\} \tag{4.6}$$

Similar to real numbers, it is possible to define additional binary operations derived from the above fundamental operations. Two of these operations are exponentiation and logarithm. Using the same sequences $\mathbf{X}$ and $\mathbf{Y}$ previously considered, we now define exponentiation and logarithm.

[1] For this operation, we decided not to use the infix notation in order to keep the number of symbols to a minimum.

$$\text{Exponentiation:} \qquad \mathbf{X}^{\mathbf{Y}} = \{(i, c[i]) : c[i] = (x[i])^{y[i]} \text{ if } x[i] \neq 0, \text{ otherwise } c[i] = 0,\ i \in \mathbb{Z}_N\} \tag{4.7}$$

$$\text{Logarithm:} \qquad \log_{\mathbf{X}} \mathbf{Y} = \{(i, c[i]) : c[i] = \log_{x[i]} (y[i]),\ i \in \mathbb{Z}_N\} \tag{4.8}$$

After defining binary operations between sequences, we open the possibility to have operations between scalars and sequences. This is because a scalar and a constant sequence behave in a similar manner when used as operands with sequences.

Definition 4 (Operations Between Scalars and Sequences)**.** *Let us consider a scalar* $k \in \mathbb{R}$ *and a sequence* $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$*, we now define the following operations between scalars and sequences.*

$$K + \mathbf{X} = \mathbf{X} + K = \{(i, c[i]) : c[i] = x[i] + K,\ i \in \mathbb{Z}_N\} \tag{4.9}$$

$$K \times \mathbf{X} = \mathbf{X} \times K = \{(i, c[i]) : c[i] = x[i] \times K,\ i \in \mathbb{Z}_N\} \tag{4.10}$$

$$\max(\mathbf{X}, K) = \max(K, \mathbf{X}) = \{(i, c[i]) : c[i] = \max(x[i], K),\ i \in \mathbb{Z}_N\} \tag{4.11}$$

$$\mathbf{X}^K = \{(i, c[i]) : c[i] = x[i]^K \text{ if } x[i] \neq 0, \text{ otherwise } c[i] = 0,\ i \in \mathbb{Z}_N\} \tag{4.12}$$

$$\log_K \mathbf{X} = \{(i, c[i]) : c[i] = \log_K x[i],\ i \in \mathbb{Z}_N\} \tag{4.13}$$

As previously noted, scalar operations can also be defined by using a constant sequence. For this, let us consider the constant sequence $\mathbf{Z} = \{(i, K) : K \in \mathbb{R} \text{ for every } i \in \mathbb{Z}_N\}$, then,

$$\mathbf{X} + K = \mathbf{X} + \mathbf{Z}$$

$$\mathbf{X} \times K = \mathbf{X} \times \mathbf{Z}$$

$$\max(\mathbf{X}, K) = \max(\mathbf{X}, \mathbf{Z})$$

$$\mathbf{X}^K = \mathbf{X}^{\mathbf{Z}}$$

$$\log_K \mathbf{X} = \log_{\mathbf{Z}} \mathbf{X}$$

Continuing with the analogy between real-valued sequences and the set of real numbers, the binary operations between sequences; subtraction ($-$), division ($\div$), and minimum (min) can be defined in terms of the previously defined fundamental operations. Let $\mathbf{X}$, $\mathbf{Y} \in \mathbb{R}^{\mathbb{Z}_N}$ be two sequences, the derived binary operations between sequences can be defined as follows.

$$\mathbf{X} - \mathbf{Y} = \mathbf{X} + (-\mathbf{Y}), \tag{4.14}$$

where $-\mathbf{Y}$ is an unary operation defined as

$$-\mathbf{Y} = \{(i, -y[i]) : (i, y[i]) \in \mathbf{Y}\}. \tag{4.15}$$

The operations of division and minimum are defined as follows.

$$\mathbf{X} \div \mathbf{Y} = \mathbf{X} \times (\mathbf{Y})^{-1} \tag{4.16}$$

$$\min(\mathbf{X}, \mathbf{Y}) = -\max(-\mathbf{X}, -\mathbf{Y}) \tag{4.17}$$

It is also possible to extend the use of unary operations from the set of real values to the set of finite real-valued sequences. However, due to the large variety of possible unary operations (trigonometric functions, ceiling, floor, absolute value, to name just a few), we define only a general unary operation over sequences.

Definition 5 (Unary Operations over Sequences)**.** *Let* $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$ *be a sequence. A unary operation over a real-valued sequence can be formally described as a function* $u : \mathbb{R}^{\mathbb{Z}_N} \to \mathbb{R}^{\mathbb{Z}_N}$ *and defined as follows.*

$$u(\mathbf{X}) = \{(i, c[i]) : c[i] = u(x[i]),\ i \in \mathbb{Z}_N\} \tag{4.18}$$

A logic operation is one that goes from $\mathbb{R}^{\mathbb{Z}_N}$ to $\{1, 0\}^{\mathbb{Z}_N}$. A logic predicate is applied to each element on a real-valued sequence, and, as a result, a sequence of ones and zeros is created. We formalize such operation in the following definition.

Definition 6 (Logic Operation over Sequences)**.** *Let* $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$ *be a sequence and* $P : \mathbb{R} \to \{1, 0\}$ *a predicate. A logic function* $B_P : \mathbb{R}^{\mathbb{Z}_N} \to \{1, 0\}^{\mathbb{Z}_N}$ *is defined as follows.*

$$B_P(\mathbf{X}) = \{(i, c[i]) : c[i] = P(x[i]),\ i \in \mathbb{Z}_N\} \tag{4.19}$$

Besides logic operations, we have to consider additional operations for reducing sequences into real numbers. Such kind of operations will be named *Aggregate Functions.*

Definition 7 (Aggregate Functions over Sequences)**.** *Let* $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$ *be a sequence. An aggregate function of sequences is derived from a binary operation between sequences. Let us consider op to be a binary operation between real values. An aggregate function* $AGG : \mathbb{R}^{\mathbb{Z}_N} \to \mathbb{R}$ *is defined as follows.*

$$AGG(\mathbf{X}) = x[0]\ op\ x[1]\ op\ x[2]\ op\ \cdots\ op\ x[N-2]\ op\ x[N-1] \tag{4.20}$$

Particularly, we consider four aggregate functions over sequences from the binary operations $+$, $\times$, $\max$, *and* $\min$, *which are defined as follows.*

$$SUM(\mathbf{X}) = \sum_{0 \le i \le N-1} \mathbf{X} = x[0] + x[1] + x[2] + \cdots + x[N-2] + x[N-1] \tag{4.21}$$

$$MUL(\mathbf{X}) = \prod_{0 \leq i \leq N-1} \mathbf{X} = x[0] \times x[1] \times x[2] \times \cdots \times x[N-2] \times x[N-1] \tag{4.22}$$

$$MAX(\mathbf{X}) = \max_{0 \leq i \leq N-1} x[i] \tag{4.23}$$

$$MIN(\mathbf{X}) = \min_{0 \leq i \leq N-1} x[i] \tag{4.24}$$

Additionally, we can obtain an aggregate function by using the cardinality of a set, $|\cdot|$. In our proposed model, the cardinality of a set representing a sequence, is equivalent to the number of elements of the sequence. Therefore, we define the aggregate function $COUNT : \mathbb{R}^{\mathbb{Z}_N} \to \mathbb{Z}^*$, with $\mathbb{Z}^* = \{0\} \cup \mathbb{Z}^+$, using the cardinality of a set as follows.

$$COUNT(\mathbf{X}) = |\mathbf{X}| \tag{4.25}$$

Other aggregate functions can be obtained combining the previous functions. For instance, the function average, $AVG : \mathbb{R}^{\mathbb{Z}_N} \to \mathbb{R}$, can be defined considering a non empty set $\mathbf{X}$ using the SUM and the $COUNT$ functions as follows.

$$AVG(\mathbf{X}) = \frac{SUM(\mathbf{X})}{COUNT(\mathbf{X})} \tag{4.26}$$

The next operation we define, *indexing transformation*, can be used to modify a sequence in terms of the indexing locations of its elements.

Definition 8 (Indexing Transformations over Sequences)**.** *Let $\mathbb{Z}_N$ and $\mathbb{Z}_M$ be two indexing sets. An indexing transformation on a sequence $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$ can be obtained by using a composition between a sequence and a function $f : \mathbb{Z}_M \to \mathbb{Z}_N$. Then, an indexing transformation is defined as*

$$\mathbf{X} \circ f = \{(j, x[f(j)] : j \in \mathbb{Z}_M\} \tag{4.27}$$

Basically, an indexing transformation receives an input sequence of length N and returns a sequence of length M. Therefore, this transformation is not only capable

of modifying the position of the elements in the sequence, but also, the length of the sequence. A particular common indexing transformation is the reflection of sequences. This operation is used when it is necessary to invert the order of the elements in a sequence. The last element of an input sequence becomes the first element of the output sequence and the rest of the elements are moved accordingly.

4.3 Signal Processing Operations

The previously defined operations and functions cover the basics of sequence transformation. Nevertheless, signal transformations often require more sophisticated operations such as those based on sliding windows. In order to include this transformation in the proposed algebra, we define an abstract object named *kernel structure*, which is a matrix-like structure that allow us to construct an sliding window as it is described in Chapter 5.

Definition 9 (Kernel Structure)**.** *Let $\mathbb{Z}_N$ and $\mathbb{Z}_M$ be two indexing sets. A kernel structure is a function $\mathscr{H} : \mathbb{Z}_M \to \mathbb{R}^{\mathbb{Z}_N}$ that maps each $j \in \mathbb{Z}_M$ into a real valued sequence $\mathscr{H}[j] \in \mathbb{R}^{\mathbb{Z}_N}$. For notation convenience, let us define $\mathbf{H}_j \equiv \mathscr{H}[j]$. Formally, each sequence $\mathbf{H}_j$ is defined as*

$$\mathbf{H}_j = \{(i, h_j[i]) : i \in \mathbb{Z}_N\} \tag{4.28}$$

The set of all real valued kernel structures from $\mathbb{Z}_M$ to $\mathbb{Z}_N$ is denoted by $(\mathbb{R}^{\mathbb{Z}_N})^{\mathbb{Z}_M}$.

The kernel structure can be understood as a template where the values of each of its sequences $\mathbf{H}_j$ are defined based on the context of their usage in a particular operation. In Section 5.1, the process of filling in or defining such values is described. A kernel structure can be graphically represented as a matrix-like structure as shown

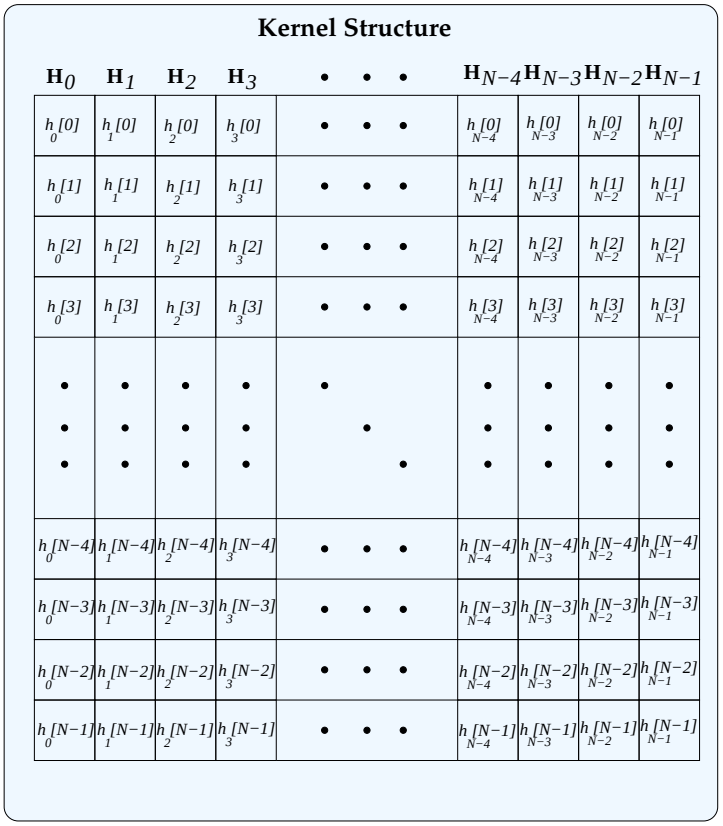

Figure 4.1: Kernel Structure

in Figure 4.1. Next, we provide a formal definition for a processing operation between a sequence and a kernel structure.

Definition 10 (Processing Operations over Sequences)**.** *Let* $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$ *be a sequence. A processing operation,* $\circledcirc$*, between a sequence and a kernel structure,* $\mathscr{H}$*, is defined by the following expression.*

$$\mathbf{X} \circledcirc \mathscr{H} = \{(j, w[j]) : w[j] = AGG(\mathbf{X} \; op \; \mathbf{H}_j), \; j \in \mathbb{Z}_M\}, \tag{4.29}$$

where $AGG \in \{SUM, MUL, MAX, MIN, COUNT\}$*, and* $op \; \in \{+, -, \times, \div\}$*.*

Some examples of processing operations are linear filtering ($\otimes$), erosion ($\ominus$), dilation ($\oplus$), and subsequence matching ($\oslash$). We can express each of these operation as

follows.

$$\mathbf{X} \otimes \mathscr{H} = \{(j, w[j]) : w[j] = \sum(\mathbf{X} \times \mathbf{H}_j),\ j \in \mathbb{Z}_N\} \tag{4.30}$$

$$\mathbf{X} \ominus \mathscr{H} = \{(j, w[j]) : w[j] = MIN(\mathbf{X} - \mathbf{H}_j),\ j \in \mathbb{Z}_N\} \tag{4.31}$$

$$\mathbf{X} \oplus \mathscr{H} = \{(j, w[j]) : w[j] = MAX(\mathbf{X} + \mathbf{H}_j),\ j \in \mathbb{Z}_N\} \tag{4.32}$$

$$\mathbf{X} \oslash \mathscr{H} = \{(j, w[j]) : w[j] = \sum((\mathbf{X} - \mathbf{H}_j)^2),\ j \in \mathbb{Z}_N\} \tag{4.33}$$

It is worth noting that $\mathbf{X}$ is a sequence of length N ($\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$) while $\mathbf{X} \circledcirc \mathscr{H}$ is a sequence of length M ($\mathbf{X} \circledcirc \mathscr{H} \in \mathbb{R}^{\mathbb{Z}_M}$). This means that a processing operation is capable of modifying the length of an input sequence. In consequence, our model has the expressive power to define other common length-changing sequence transformations, such as *decimation* and *interpolation*.

Chapter 5
Modeling Analysis Algorithms: ECG Signals

In this chapter, we use our proposed signal algebra to express ECG signal processing techniques. We express a particular example of QRS wave detection using signal algebra. Moreover, we prove that the proposed model is capable of expressing any transformation from a finite digital signal into another finite digital signal.

5.1 Modeling ECG Analysis

We present five techniques; linear filtering, morphological operators, subsequence matching, discrete Fourier transform, and SAX. We have found that all these techniques are similar in their operation in the sense that all of them are based on an *sliding window*.

5.1.1 Sliding Window

The sliding window is a procedure that allows a window of specific length to move through a sequence. In doing so, it defines subsequences along the path. The sliding window procedure is illustrated in Figure 5.1, where a window of length 4 slides along sequence $\mathbf{X}$. At each position i of sequence $\mathbf{X}$, a subsequence of length 4 is defined by the window. In a sequence processing operation, each subsequence is transformed, usually, into a single value, which in turn becomes an element of a resulting sequence $\mathbf{Z}$. This transformation is usually, but not limited to, a two steps process. First, each subsequence of $\mathbf{X}$ is transformed by applying a binary operation (op) with a sequence

that characterizes the whole processing operation (e.g., a FIR filter); the sequence $\mathbf{Y}$ in this case. Second, the transformed subsequence $\mathbf{P}_i$ is reduced to a single value by an aggregate function to obtain an output sequence as shown in Figure 5.2.

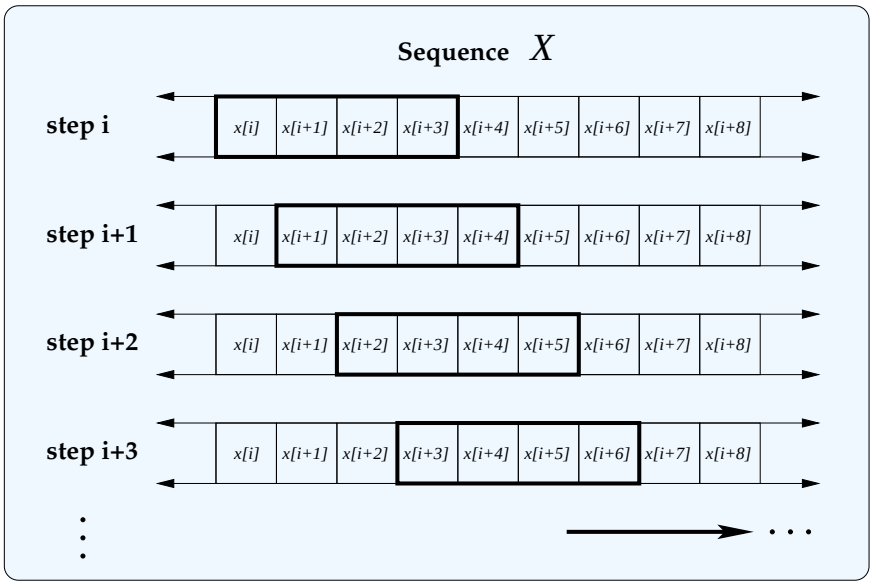

Figure 5.1: Sliding Window.

5.1.2 Linear Filtering

Linear filtering is widely used for processing ECG signals. It is accomplished by applying the convolution operation between a signal and a filter's impulse response sequence. Common processing applications such as matched and adaptive filtering make use of Finite Impulse Response (FIR) filters [72].

In a FIR filter, the convolution is computed using Equation (2.8). In order perform this computation, we need to pass an sliding window along the input sequence and perform a sequence operation at each position. This process is equivalent to the task of dividing the input sequence into overlapped subsequences of the same length of

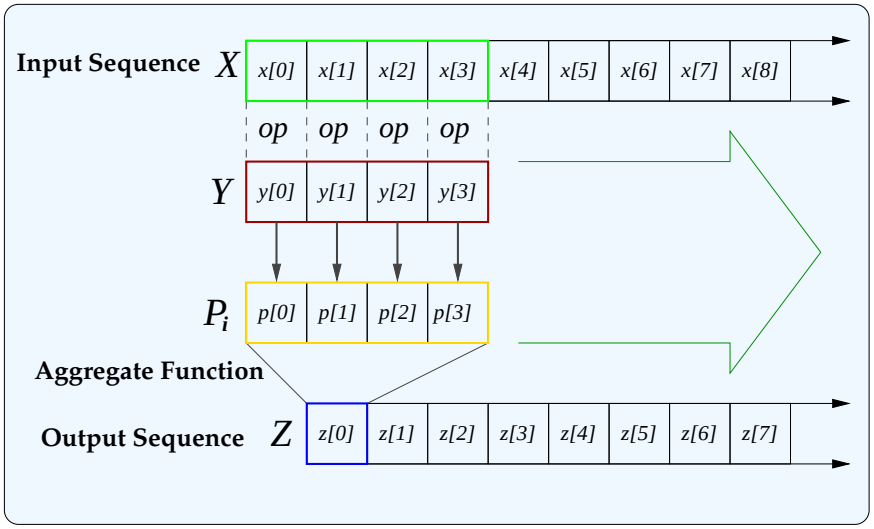

Figure 5.2: Processing operation on a sequence.

the impulse response sequence and then applying sequence operations between the individual subsequences and the impulse response sequence.

Let $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{\mathbb{Z}_N}$ be an input sequence and the impulse response sequence of a FIR filter, respectively. As part of the convolution operation, the impulse response sequence is to be reflected. We can implement this reflection by building a kernel structure in such a manner that the reflected impulse response sequence will be shifted at each element of the kernel structure. This allows us to use each $\mathbf{H}_j$ and multiply it by the input sequence defining each step of an sliding window. After this element-wise multiplication, an aggregate function reduces the resulting sequence into a scalar which will be an element of the output sequence.

Following the notation used in Definition 9, we construct the kernel structure $\mathscr{H} : \mathbb{Z}_N \to \mathbb{R}^{\mathbb{Z}_N}$ by defining its sequence elements $\mathbf{H}_j$ ($\mathbf{H}_j \equiv \mathscr{H}[j]$) as follows.

$$
\begin{aligned}
&\mathbf{H}_0 = \{(0, y[0]), (1, 0), (2, 0), \ldots, (N-2, 0), (N-1, 0)\},\\
&\mathbf{H}_1 = \{(0, y[1]), (1, y[0]), (2, 0), \ldots, (N-2, 0), (N-1, 0)\},\\
&\vdots\\
&\mathbf{H}_{N-1} = \{(0, y[N-1]), (1, y[N-2]), \ldots, (N-2, y[1]), (N-1, y[0])\}.
\end{aligned}
$$

Alternatively, we can describe each sequence $\mathbf{H}_j$ as

$$\mathbf{H}_j = \{(i, h_j[i]) : h_j[i] = y[-i+j], \text{ with } h_j[i] = 0 \text{ if } (-i+j) \notin \mathbb{Z}_N\}.$$

Figure 5.3 shows the graphic representation of the kernel structure $\mathscr{H}$. As we can see, the kernel structure defines an sliding window with the position of its elements. Finally, a FIR filtering ($\otimes$) of a sequence $\mathbf{X}$ with an impulse response sequence $\mathbf{Y}$, both of length N, can be obtained by the following expression.

$$\mathbf{X} \otimes \mathscr{H} = \{(j, w[j]) : w[j] = SUM(\mathbf{X} \times \mathbf{H}_j),\ j \in \mathbb{Z}_N\} \tag{5.1}$$

Figure 5.3 shows the graphic representation of the kernel structure $\mathscr{H}$.

Note that, for ease of explanation, we built the kernel structure in such a manner that $|\mathbf{X} \otimes \mathscr{H}| = N$ despite the fact the length of the convolution operation between two sequences of length of N is $2N-1$. However, the length of the sequence $\mathbf{x} \otimes \mathscr{H}$ can be defined as required in a particular application by building a suitable kernel structure.

Our proposed model provides a formal expression for FIR filtering. Expression (5.1) represents one of the most utilized processing algorithms for digital signals. Moreover, this expression can be used for implementing the dyadic discrete wavelet transforms [30, 49] and cubic spline interpolation [24, 97] as they can be expressed by FIR filtering. Both techniques have been widely used to address ECG analysis tasks such as signal enhancement and feature extraction, as described in Chapter 2.

5.1.3 Mathematical Morphology

Mathematical Morphology (MM) operators also use an sliding window for processing signals. Each subsequence defined by the sliding window interacts with a structuring element (SE). In the case of erosion, each subsequence is added to the SE sequence,

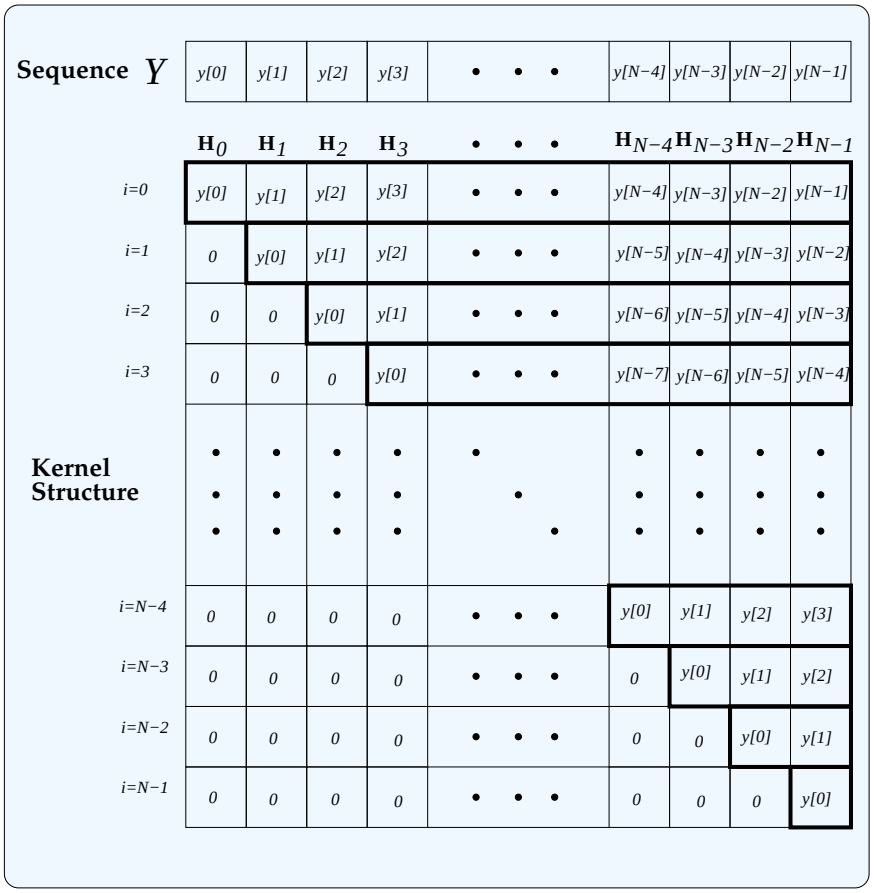

Figure 5.3: Kernel Structure $\mathscr{H}$ for digital filtering.

element-wise. On the other hand, for dilation, the reflected SE sequence and each subsequence are subtracted element-wise. Finally, the operations of minimum and maximum are applied to each resulting subsequence to obtain each element of the output sequence.

Let $\mathbf{Y} \in \mathbb{R}^{\mathbb{Z}_N}$ be a structuring element sequence for processing an input sequence $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$. We can build a kernel structure $\mathscr{H}_{\mathbf{e}} : \mathbb{Z}_N \to \mathbb{R}^{\mathbb{Z}_N}$ for the erosion operation. Again, the construction of $\mathscr{H}_{\mathbf{e}}$ allows us to represent an sliding window. Each $\mathbf{H}_{\mathbf{e}_j}$ is subtracted to the input sequence and then the minimum element of the resulting sequence is obtained. We define the elements of $\mathscr{H}_{\mathbf{e}}$, that is, the sequences $\mathbf{H}_{\mathbf{e}_j}$

($\mathbf{H}_{\mathbf{e}j} \equiv \mathscr{H}_{\mathbf{e}}[j]$) as follows.

$$\begin{aligned}
&\mathbf{H}_{\mathbf{e}0} = \{(0, y[N-1]), (1,0), (2,0), \ldots, (N-2,0), (N-1,0)\},\\
&\mathbf{H}_{\mathbf{e}1} = \{(0, y[N-2]), (1, y[N-1]), (2,0), \ldots, (N-2,0), (N-1,0)\},\\
&\vdots\\
&\mathbf{H}_{\mathbf{e}N-1} = \{(0, y[0]), (1, y[1]), \ldots, (N-2, y[N-2]), (N-1, y[N-1])\}.
\end{aligned}$$

A graphic representation of $\mathscr{H}_{\mathbf{e}}$ is shown in Figure 5.4. As we can see, the kernel structure $\mathscr{H}_{\mathbf{e}}$ allows us to construct an sliding window.

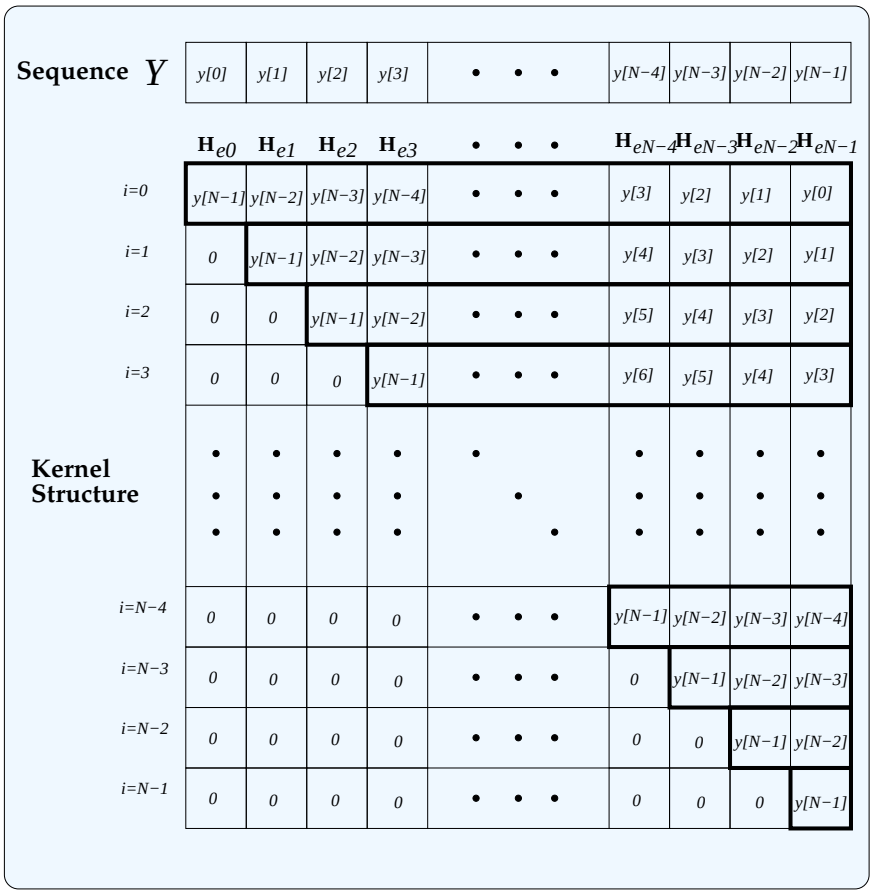

Figure 5.4: Kernel Structure $\mathscr{H}_{\mathbf{e}}$ for erosion.

Alternatively, we can express each $\mathbf{H}_{\mathbf{e}j}$ as

$$\mathbf{H}_{\mathbf{e}j} = \{(i, h_{ej}[i]) : h_{ej}[i] = y[i-j+N-1] \text{ with } h_{ej}[i] = 0 \text{ if } (i-j+N-1) \notin \mathbb{Z}_N,\ i, j \in \mathbb{Z}_N\}.$$

The MM operation erosion ($\ominus$), from Equation (2.17), can be expressed using our proposed model by the following expression.

$$\mathbf{X} \ominus \mathscr{H}_{\mathbf{e}} = \{(j, w[j]) : w[j] = MIN(\mathbf{X} - \mathbf{H}_{\mathbf{e}j}),\ j \in \mathbb{Z}_N\} \tag{5.2}$$

In the case of dilation, we can also construct a kernel structure $\mathscr{H}_{\mathbf{d}} : \mathbb{Z}_N \rightarrow \mathbb{R}^{\mathbb{Z}_N}$. The elements of $\mathscr{H}_{\mathbf{d}}$ are the sequences $\mathbf{H}_{\mathbf{d}j}$. These sequences are added element-wise to the input sequence using a binary operation. Then, the maximum element of the resulting sequence is obtained and it will be an element of the output sequence. The sequences $\mathbf{H}_{\mathbf{d}j}$ ($\mathbf{H}_{\mathbf{d}j} \equiv \mathscr{H}_{\mathbf{d}}[j]$) are defined as follows.

$$\begin{aligned}
&\mathbf{H}_{\mathbf{d}0} = \{(0, y[0]), (1, 0), (2, 0), \ldots, (N-2, 0), (N-1, 0)\},\\
&\mathbf{H}_{\mathbf{d}1} = \{(0, y[1]), (1, y[0]), (2, 0), \ldots, (N-2, 0), (N-1, 0)\},\\
&\vdots\\
&\mathbf{H}_{\mathbf{d}N-1} = \{(0, y[N-1]), (1, y[N-2]), \ldots, (N-2, y[1]), (N-1, y[0])\}.
\end{aligned}$$

The graphic representation of $\mathscr{H}_{\mathbf{d}}$ is similar to the kernel structure for linear filtering shown in Figure 5.3.

Likewise, an alternative expression for the sequences $\mathbf{H}_{\mathbf{d}j}$ is given by

$$\mathbf{H}_{\mathbf{d}j} = \{(i, h_{dj}[i]) : h_{dj}[i] = y[-i + j] \text{ with } h_{dj}[i] = 0 \text{ if } (-i + j) \notin \mathbb{Z}_N\}.$$

After this, the MM operation dilation ($\oplus$) which is defined in Equation (2.18) can be represented using the following expression.

$$\mathbf{X} \oplus \mathscr{H}_{\mathbf{d}} = \{(j, w[j]) : w[j] = MAX(\mathbf{X} + \mathbf{H}_{\mathbf{d}j}),\ j \in \mathbb{Z}_N\} \tag{5.3}$$

5.1.4 Subsequence Matching

Subsequence matching can also be understood as an sliding window procedure. Here, a pattern sequence is compared to each subsequence in the input sequence using a distance function. The pattern sequence and each subsequence are of the same length.

The process of defining the subsequences at every position of the input sequence is accomplished by an sliding window. Then, a distance value is computed at every position step of the sliding window. Consequently, a resulting sequence is obtained and its elements are the distances computed between the pattern sequence and each subsequence of the input sequence.

Let us consider an example of similarity search using the $1NN$ (1 nearest neighbor) approach. Let $\mathbf{Y} \in \mathbb{R}^{\mathbb{Z}_N}$ be a beat pattern sequence which is required to be searched along an input sequence $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_M}$ with $M >> N$. We need to construct a kernel structure $\mathscr{H}_{\mathbf{s}}$ in a way that it allows us to generate an sliding window. We can build $\mathscr{H}_{\mathbf{s}}$ by defining its elements $\mathbf{H}_{\mathbf{s}j}$ ($\mathbf{H}_{\mathbf{s}j} \equiv \mathscr{H}_{\mathbf{s}}[j]$) using the elements of the sequence $\mathbf{Y}$. Each sequence $\mathbf{H}_{\mathbf{s}j}$ represents a step that the beat pattern travels along the input sequence. In this way, we can apply a distance function at every step in order to find the searched pattern. We define $\mathbf{H}_{\mathbf{s}j}$ as follows.

$$
\begin{aligned}
&\mathbf{H}_{\mathbf{s}0} = \{(0, y[0]), (1, y[1]), \ldots, (N-1, y[N-1]), \ldots, (M-2, 0), (M-1, 0)\},\\
&\mathbf{H}_{\mathbf{s}1} = \{(0, 0), (1, y[0]), \ldots, (N-1, y[N-2]), (N, y[N-1]), \ldots, (M-1, 0)\},\\
&\vdots\\
&\mathbf{H}_{\mathbf{s}N-1} = \{(0, 0), (1, 0), \ldots, (N-1, 0), (N, y[0]), \ldots, (M-1, y[N-1])\}.
\end{aligned}
$$

The sequences $\mathbf{H}_{\mathbf{s}j}$ are given by the following expression.

$$\mathbf{H}_{\mathbf{s}j} = \{(i, h_{sj}[i]) : h_{sj}[i] = y[i-j] \text{ with } h_{sj}[i] = 0 \text{ if } (i-j) \notin \mathbb{Z}_N\}$$

The graphic representation of $\mathscr{H}_{\mathbf{s}}$ is shown in Figure 5.5.

The expression for the computation of the sequence distance ($\oslash$) is obtained using a processing operation with a square exponentiation operation. Here, we also use a logic function $B_{\neq 0}$ to keep only M non zero values before applying the aggregation operation SUM. The rest of the elements will be zeros. In this case, the logic output

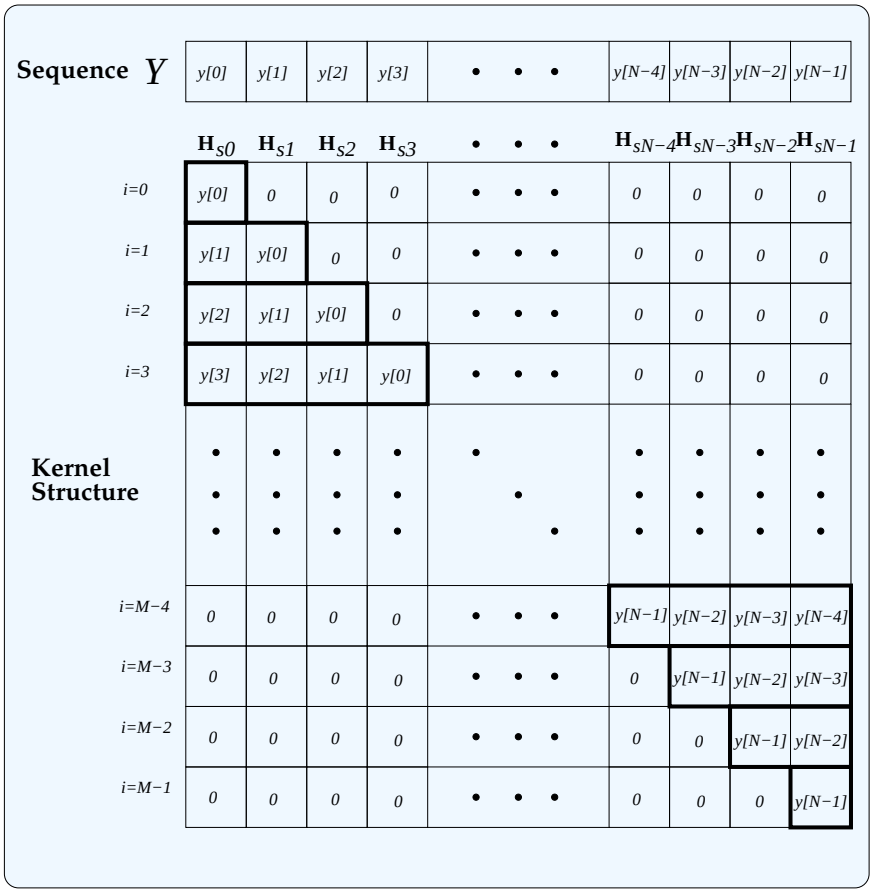

Figure 5.5: Kernel Structure $\mathscr{H}_{\mathbf{s}}$ for subsequence matching.

values from the logic function are 1 and 0. We compute the sequence of distance by using the following expression of our proposed model.

$$(\mathbf{X} \oslash \mathscr{H}_{\mathbf{s}})^{\frac{1}{2}} = \{(j, w[j]) : w[j] = \left(SUM\left((\mathbf{X} - \mathbf{H}_{\mathbf{s}_j})^2 \times B_{\neq 0}(\mathbf{H}_{\mathbf{s}_j})\right)\right)^{\frac{1}{2}}, j \in \mathbb{Z}_N\} \quad (5.4)$$

The values at each position of the sequence $\mathbf{X} \oslash \mathscr{H}_{\mathbf{s}}$ are the distances between the sequence pattern and the subsequence at that position. The last step is finding the position of the minimum element of the sequence $\mathbf{X} \oslash \mathscr{H}_{\mathbf{s}}$), that is, the position of the value $MIN(\mathbf{X} \oslash \mathscr{H}_{\mathbf{s}})$. Expression (5.4) represents the operation sub-

sequence matching using Euclidean distance. In the literature we can find several approaches [23, 36, 46] that can also be defined using this expression.

5.1.5 Discrete Fourier Transform

In order to express the real valued DFT using our proposed model, we build two kernel structures, namely, $\mathscr{H}_{\mathbf{a}}$ and $\mathscr{H}_{\mathbf{b}}$. If we consider $\mathbf{X}$ as the input sequence, the formulation of the real DFT uses the linear filtering operation($\otimes$) and it is given by

$$\mathbf{X_1} = (\mathbf{X} \otimes \mathscr{H}_{\mathbf{a}}), \tag{5.5}$$

$$\mathbf{X_0} = (\mathbf{X} \otimes \mathscr{H}_{\mathbf{b}}), \tag{5.6}$$

where

$$\mathbf{H}_{\mathbf{a}j} = \{(i, h_{aj}[i]) : h_{aj}[i] = \cos(2\pi ji/N) \text{ for } N = |\mathbf{X}| \text{ and } i \in \mathbb{Z}_N\},$$

for $0 \leq j \leq N_1$ and $\mathbf{H}_{\mathbf{a}j} \equiv \mathscr{H}_{\mathbf{a}}[j]$ and

$$\mathbf{H}_{\mathbf{b}m} = \{(i, h_{bj}[i]) : h_{bj}[i] = \sin(2\pi mi/N) \text{ for } N = |\mathbf{X}| \text{ and } i \in \mathbb{Z}_N\},$$

for $1 \leq m \leq N_0$, and $\mathbf{H}_{\mathbf{b}j} \equiv \mathscr{H}_{\mathbf{b}}[j]$.

As we can see in Expressions (5.5) and (5.6), we can express the real Discrete Fourier Transform using our proposed model with one processing operation.

5.1.6 *SAX* Approximation of a Signal

The transformation of a digital signal into a symbolic representation have been considered by researchers in order to take advantage of text processing and bioinformatics algorithms. A popular choice of symbolic representation is the Symbolic Aggregate Approximation (SAX) proposed by Lin et al. [47]. To obtain the SAX representation

of a signal, we start by normalizing and transforming the original signal using Piecewise Aggregate Approximation (PAA). This first step generates an output sequence that now, compared to the original sequence, is shorter, its mean is equal to zero, and its standard deviation is equal to one. The second step of the SAX process consists of mapping the PAA coefficients into SAX symbols. The SAX symbols have the property of equiprobability. The mapping procedure makes use of subsets of the real numbers that have a SAX symbol associated. Then, each PAA value is compared to the subsets in order to assign a SAX symbol.

Let us consider an input sequence $\mathbf{X}$ of length N. The process of transforming $\mathbf{X}$ into a PAA sequence of length W (assuming $N \bmod W = 0$) can be defined in our model by the Expression (5.7).

$$\mathbf{X_{PAA}} = (\mathbf{X} \otimes \mathscr{H}_{\mathbf{x}}), \tag{5.7}$$

where

$$\mathbf{H}_{\mathbf{x}j} = \{(i, h_{aj}[i]) : h_{aj}[i] = \frac{W}{N} \text{ if } j\frac{N}{W} \leq i \leq (j{+}1)\frac{N}{W}{-}1 \text{ otherwise } h_{aj}[i] = 0,\ i \in \mathbb{Z}_N\}$$

with $0 \leq j \leq W-1$ and $\mathbf{H}_{\mathbf{x}j} \equiv \mathscr{H}_{\mathbf{x}}[j]$. That is, each sequence $\mathbf{H}_{\mathbf{x}j}$ allows us to divide the input sequence into W frames and to compute the average at each frame. Finally, each SAX symbol can be assigned by using a logic operation as described in Definition 6. Using our proposed model, a SAX symbol associated to the set $\mathbf{S}$ will be assigned by using the Expression (5.8).

$$B_{\mathbf{S_m}}(\mathbf{X_{PAA}}) = \{(i, c[i]) : c[i] = T \text{ if } x_{PAA}[i] \in \mathbf{S_m}, \text{ otherwise } c[i] = F,\ i \in \mathbb{Z}_W\} \tag{5.8}$$

A true value in the sequence $T_{\mathbf{S_m}}(\mathbf{X_{PAA}})$ will indicate that the SAX symbol associated to the set $\mathbf{S_m}$ will be assigned in that position. The normalization was not

described in the previous equations but it can be expressed using the scalar operations that were described in Definition 4.

5.2 Signal Processing Example: QRS Detection

One of the most popular QRS detection method is the one proposed by Pan and Tompkins [67], which uses an algorithmic structure similar to the one illustrated in Figure 2.6. The preprocessing stage of this algorithm starts with a band-pass filter. This filter is implemented through a low-pass filter (LPF) and a high-pass filter (HPF) in cascade, filtering out all frequencies outside of the range 5-12 Hz. The filtered signal is then differentiated, squared and averaged. Differentiation provides slope information while the squaring process makes all sequence's values positive and performs nonlinear amplification emphasizing the higher frequencies [67]. Time averaging or moving window integration consists in adding the last C samples and dividing by C. Figure 5.6 shows a simplified flow diagram of the Pan-Tompkins QRS complex detector. This algorithm uses two sets of thresholds, one for the moving window integration sequence and another one for the filtered ECG sequence. This improves the reliability of QRS complex detection compared to using one sequence alone [67]. In order to be identified as a QRS complex, a peak must be recognized as such on both sequences.

Low and high linear filtering, differentiation and time averaging can be obtained by linear filtering. Each of these processes requires a kernel structure characterized by the corresponding sequence. For the particular case of the Pan-Tompkins method, the sequences that characterizes the processes are depicted in Figure 5.7. The kernel structures built using such sequence are denoted as $\mathscr{H}_{\mathbf{L}}$ (low pass filter), $\mathscr{H}_{\mathbf{H}}$ (high pass filter), $\mathscr{H}_{\mathbf{D}}$ (derivative), and $\mathscr{H}_{\mathbf{A}}$ (average filter), respectively.

Expression (5.9) is the formal representation of the preprocessing stage of the

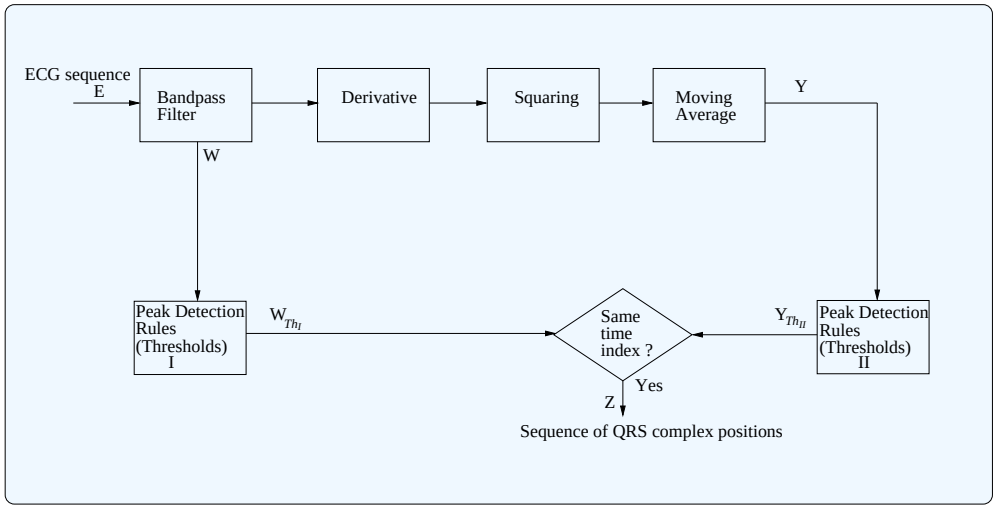

Figure 5.6: Flow diagram of the Pan-Tompkins algorithm.

Pan-Tompkins algorithm, where $\mathbf{Y}$ is the output and $\mathbf{E}$ represents the original ECG sequence.

$$\mathbf{Y} = (((\mathbf{E} \otimes \mathscr{H}_{\mathbf{L}}) \otimes \mathscr{H}_{\mathbf{H}}) \otimes \mathscr{H}_{\mathbf{D}})^2 \otimes \mathscr{H}_{\mathbf{A}} \tag{5.9}$$

and the filtered sequence $\mathbf{W}$ can be formally expressed as

$$\mathbf{W} = ((\mathbf{E} \otimes \mathscr{H}_{\mathbf{L}}) \otimes \mathscr{H}_{\mathbf{H}}) \tag{5.10}$$

The process of peak detection consists of determining the positions where the signal changes direction within a predefined time interval. In the decision process, each peak, a local maximum, is classified as a signal peak or noise using two sets of thresholds with two threshold levels in each of the sets. These threshold levels are continuously modified and adapted to the most recent characteristics of the signal. However, since in our case, we are representing stored signals, we use a simpler scheme of thresholds based on the scheme of the Pan-Tompkins method, originally designed for on-line detection. In Expressions (5.11) and (5.12), we define two thresholds denoted as Th_I

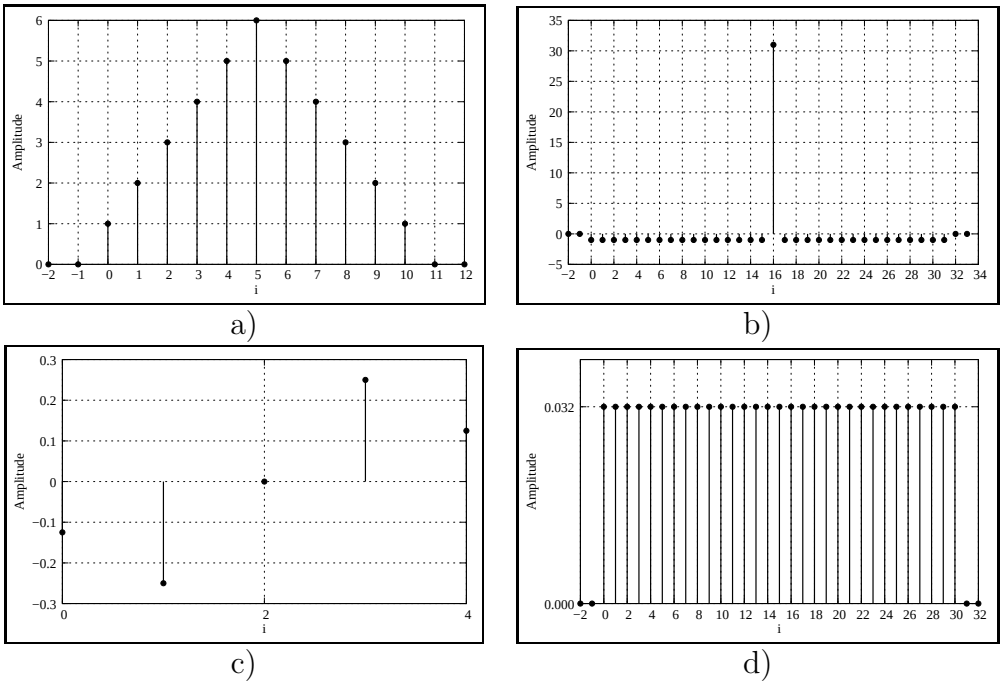

Figure 5.7: Sequences that characterize a) Low pass filter, b) High pass filter, c) Derivative filter and d) Average filter.

and Th_{II} to be applied to sequences $\mathbf{W}$ and $\mathbf{Y}$, respectively.

$$Th_I = NPKF + 0.25(SPKF - NPKF), \tag{5.11}$$

where $NPKF$ is an estimate of the noise peak of the signal $\mathbf{W}$ and $SPKF$ is an estimate of the signal peak of $\mathbf{W}$.

$$Th_{II} = NPKI + 0.25(SPKI - NPKI), \tag{5.12}$$

where $NPKI$ is an estimate of the noise peak of the signal $\mathbf{Y}$ and $SPKI$ is an estimate of the signal peak of $\mathbf{Y}$.

Most of the decision stages of the QRS detection algorithms are rather heuristic and dependent on the results of the preprocessing stage [39]. Therefore, we represent peak detection using the logic operations $B_{>Th_I}$ and $B_{>Th_{II}}$. Logic operations were

described in Definition 6. We denote $\mathbf{W}_{Th_I}$ and $\mathbf{Y}_{Th_{II}}$ as the sequences obtained from the peak detection process. The following expression are the representation of these sequences.

$$\mathbf{W}_{Th_I} = B_{>Th_I}(\mathbf{W}) \tag{5.13}$$

$$\mathbf{Y}_{Th_{II}} = B_{>Th_{II}}(\mathbf{Y}) \tag{5.14}$$

Finally, the position of the QRS waves is determined where both sequences, $\mathbf{W}_{Th_I}$ and $\mathbf{Y}_{Th_{II}}$, coincide. This requires a sequence logic operation "AND" which can be expressed using the binary operation min(). Expression (5.15) represents the final decision stage.

$$\mathbf{Z} = \min(\mathbf{W}_{Th_I}, \mathbf{Y}_{Th_{II}}) \tag{5.15}$$

The sequence $\mathbf{Z}$ contains an element of value "1" at each position where the QRS wave is located.

5.3 Sufficiency of the Proposed Model

In this section, we prove that our proposed model is capable of expressing any transformation from a finite digital signal into another finite digital signal. We use the approach described in the work of Ritter et al. [75], where a proof of sufficiency of an image algebra was presented. We start with some preliminaries that help us in the development of the proof.

Let us define a sequence $\mathbf{B_k} = \{(i, b_k[i]) : i \in \mathbb{Z}_N\}$ for each $k \in \mathbb{Z}_N$, where the values $b_k[i]$ are defined as follows.

$$b_k[i] = \begin{cases} 1, & \text{when } k = i, \\ 0, & \text{otherwise.} \end{cases}$$

A graphic representation of sequences $\mathbf{B_k}$ is shown in Figure 5.8. The values at positions where $k = i$ are one and zero elsewhere.

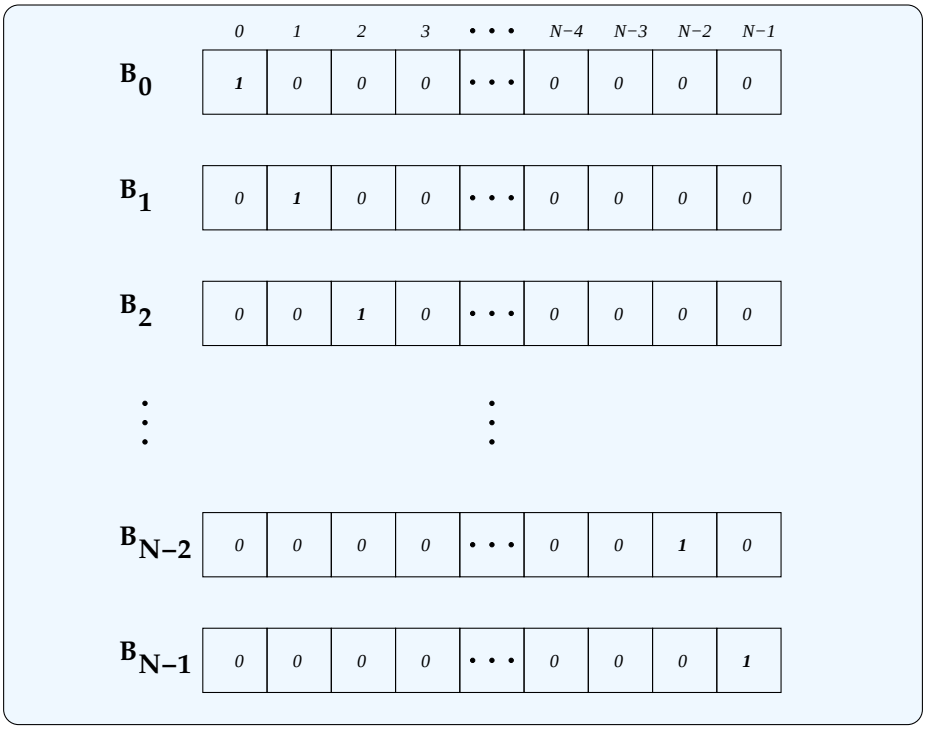

Figure 5.8: Sequences $\mathbf{B_k}$ where $b_k[i] = 1$ for $k = i$.

Let $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}$ be an arbitrary sequence and $f(y_0, y_1, \ldots, y_{N-1})$ be a multivariate polynomial with real coefficients and real variables. Then, let us consider the function $\bar{f}(\cdot)$, a sequence version of the polynomial $f(\cdot)$, where now the variables are real valued sequences and the operations involved in the polynomial are sequence operations, as they are defined in the proposed model . The multivariate sequence polynomial, $\bar{f}(\cdot)$, can be represented using two equivalent expressions as follows.

$$\bar{f}(\mathbf{X}) \equiv \bar{f}(SUM(\mathbf{X} \times \mathbf{B_0}) \times \mathbf{1}, SUM(\mathbf{X} \times \mathbf{B_1}) \times \mathbf{1}, \ldots, SUM(\mathbf{X} \times \mathbf{B_{N-1}}) \times \mathbf{1})$$

We construct the variables for $\bar{f}(\cdot)$ by defining the sequences $SUM(\mathbf{X} \times \mathbf{B_k}) \times \mathbf{1}$ with all its elements being the same variable element. That is, each sequence $SUM(\mathbf{X} \times \mathbf{B_k}) \times \mathbf{1}$ is a constant sequence with all values equal $x[k]$. For example, variable

$SUM(\mathbf{X} \times \mathbf{B_1}) \times \mathbf{1}$ is the sequence where all its elements are the element $x[1]$ of sequence $\mathbf{X}$, as represented in the following expression.

$$SUM(\mathbf{X} \times \mathbf{B_1}) \times \mathbf{1} = (x[1], x[1], x[1], \ldots, x[1], x[1], x[1])$$

where $|SUM(\mathbf{X} \times \mathbf{B_1}) \times \mathbf{1}| = N$.

Basically, the multivariate polynomial function $\bar{f}(\mathbf{X})$ where its variables take values from the sequences $\mathbf{X}$. The number of variables is equal to the length of $\mathbf{X}$, in this case, $|\mathbf{X}| = N$. That is, N constant sequences are formed using each element of $\mathbf{X}$ as the values of the sequences.

Since the proposed model is intended to represent sequences stored in computer systems, the set of possible real values taken by the elements of any of stored sequence is finite. The cardinality of this set depends on the number of bits used to represent each value. We define the set of possible real numbers taken by a sequence as $\mathbb{G} = \{g_1, \ldots, g_p\} \subset \mathbb{R}$. Let us consider the set of all sequences of length N with values taken from the set $\mathbb{G}$, namely, $\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}] = \{\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N} : (i, x[i]) \in \mathbf{X}, x[i] \in \mathbb{G}, i \in \mathbb{Z}_N\}$. Note that $|\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]| = p^N$.

Now, using Lagrange interpolation polynomials, we define a real univariate polynomial $h_j(x)$ for each element of $\mathbb{G}$. These polynomials are defined in such a way that $h_j(g_j) = 1$ when evaluated at a particular element of the set $\mathbb{G}$. Formally, we define those p polynomials in Lemma 1.

Lemma 1. *[Lagrange Interpolation Polynomials] For every $j \in \{1, 2, \ldots, p\}$, where $p = |\mathbb{G}|$, there is a polynomial $h_j(x)$ such that*

$$h_j(g_i) = \begin{cases} 1, & \textit{when } i = j, \\ 0, & \textit{otherwise.} \end{cases}$$

Proof. Let $L_j(x) = (x-g_1)(x-g_2)\cdots(x-g_{j-1})(x-g_{j+1})\cdots(x-g_p)$. Therefore, $L_j(g_i) = 0$ when $i \neq j$ and $L_j(g_i) \neq 0$ when $i = j$.

Let us define

$$h_j(x) = \frac{L_j(x)}{L_j(g_j)}$$

In this way, the polynomial $h_j(x)$ satisfies the conditions established in their definition.

Q.E.D.

Note that the polynomial $L_j(x)$ is defined as a multiplication of factors, where the factor $(x - g_j)$ is not included in the multiplication. Moreover, $L_j(g_j)$ is the polynomial $L_j(x)$ evaluated at g_j, thus, it is a constant real value. This allows us to define the polynomial $h_j(x)$ as a division between $L_j(x)$ and $L_j(g_j)$.

The previous definitions allow us to propose Theorem 1. Theorem 1 states that our proposed signal algebra is capable of expressing any sequence transformation from a sequence in $\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$ to a sequence in $\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$.

Theorem 1. *Let $\alpha : \mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}] \to \mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$ be any sequence transformation. Then, there is a sequence expression γ in the proposed signal algebra such that $\alpha(\mathbf{X}) = \gamma(\mathbf{X})$ for every $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$.*

Proof. Given two sequences $\mathbf{X_i}, \mathbf{X_k} \in \mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$ such that $\mathbf{X_i} \neq \mathbf{X_k}$, then, at least in one position of those sequences, the elements of the sequences are different. Formally, $x_i[r] \neq x_k[r]$ for some $r \in \mathbb{Z}_N$. Therefore, if $x_k[r] = g_j$ for some j, from Lemma 1, $h_j(x_i[r]) = 0$ and $h_j(x_k[r]) = 1$.

Considering that $|\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]| = p^N$, we define a polynomial function $f_k(\cdot)$ for each sequence in $\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$. Such polynomials are defined in such a way that $f_k(y_0, \ldots, y_{N-1}) = 1$ when evaluated at the elements of a particular sequence. Note that, the variables take the following values $y_0 = x_k[0], y_1 = x_k[1], \ldots, y_{N-2} = x_k[N-2], y_{N-1} =$

$x_k[N-1]$,. Formally, for $k = 1, \ldots, m = p^N$, let $f_k(y_0, y_1, \ldots, y_{N-1})$ be a multivariate polynomial with real variables and real coefficients as previously defined. We choose the sequence of integers $j_0, j_1, \ldots, j_{N-1}$ in such a way that $g_{j_0} = x_k[0], g_{j_1} = x_k[1], \ldots, g_{j_{N-1}} = x_k[N-1]$. Let each polynomial function be defined as follows.

$$f_k(y_0, y_1, \ldots, y_{N-1}) = h_{j_0}(y_0)h_{j_1}(y_1)\cdots h_{j_{N-1}}(y_{N-1})$$

Please note that when $i = k$, that is, when $f_k(\cdot)$ is evaluated using the elements of its corresponding sequence,

$$\begin{aligned} f_k(x_i[0], x_i[1], \ldots, x_i[N-1]) &= h_{j_0}(x_i[0])h_{j_1}(x_i[1])\cdots h_{j_{N-1}}(x_i[N-1]) \\ &= h_{j_0}(g_{j_0})h_{j_1}(g_{j_1})\cdots h_{j_{N-1}}(g_{j_{N-1}}) \\ &= 1. \end{aligned}$$

Since $h_{j_0}(g_{j_0}) = 1$, $h_{j_1}(g_{j_1}) = 1, \ldots, h_{j_{N-1}}(g_{j_{N-1}}) = 1$.

For $i \neq k$, we have that $h_j(x_i[r]) = 0$ for some j and some r. Consequently,

$$f_k(x_i[0], x_i[1], \ldots, x_i[N-1]) = 0.$$

Likewise, if we consider the previously defined sequence polynomial function $\bar{f}_k$ evaluated at some sequence $\mathbf{X_i}$, we have,

$$\bar{f}_k(\mathbf{X_i}) = \begin{cases} \mathbf{1}, & \text{when } i = k, \\ \mathbf{0}, & \text{when } i \neq k. \end{cases}$$

Note that $\mathbf{1}$ is the sequence where all the elements are ones and $\mathbf{0}$ is the sequence with only zeros as elements. Naturally, as $\bar{f}_k$ is the sequence polynomial version of f_k, $\bar{f}_k$ requires for its definition a sequence version of the polynomial $h_j(x)$, that is, $\bar{h}_j(\mathbf{X})$.

Using the operations from our proposed signal algebra, the sequence polynomial function $\bar{f}_k$ is defined as follows.

$$\bar{f}_k(\mathbf{X}) = \bar{h}_{j_0}\left(SUM(\mathbf{X} \times \mathbf{B_0}) \times \mathbf{1}\right) \times \cdots \times \bar{h}_{j_{N-1}}(SUM(\mathbf{X} \times \mathbf{B_{N-1}}) \times \mathbf{1}), \quad (5.16)$$

where

$$\bar{h}_j(\mathbf{X}) = \bar{L}_j(\mathbf{X}) \div \bar{L}_j\left(g_j \times \mathbf{1}\right), \tag{5.17}$$

with

$$\bar{L}_j(\mathbf{X}) = (\mathbf{X} - g_1 \times \mathbf{1}) \times (\mathbf{X} - g_2 \times \mathbf{1}) \times \cdots \times (\mathbf{X} - g_{j-1} \times \mathbf{1}) \times (\mathbf{X} - g_{j+1} \times \mathbf{1}) \cdots (\mathbf{X} - g_p \times \mathbf{1}). \tag{5.18}$$

Now, let the sequences $\mathbf{D_1}, \mathbf{D_2}, \ldots, \mathbf{D_m}$ be defined by $\alpha(\mathbf{X_k}) = \mathbf{D_k}$. That is, each sequence $\mathbf{X_k}$ is transformed into a sequence $\mathbf{D_k}$ by the sequence transformation α. Then, we define the expression of our proposed signal algebra γ as

$$\gamma(\mathbf{X}) = (\bar{f}_1(\mathbf{X}) \times \mathbf{D_1}) + (\bar{f}_2(\mathbf{X}) \times \mathbf{D_2}) + \cdots + (\bar{f}_m(\mathbf{X}) \times \mathbf{D_m}).$$

Then, $\gamma(\mathbf{X_k}) = \mathbf{1} \times \mathbf{D_k} = \mathbf{D_k}$ and thus, $\gamma(\mathbf{X}) = \alpha(\mathbf{X})$ for all $\mathbf{X} \in \mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$.

In this way, we state that a transformation γ exists in our model that is equivalent to α for any given sequence in $\mathbb{R}^{\mathbb{Z}_N}[\mathbb{G}]$.

Q.E.D.

Theorem 1 proves that any arbitrary transformation of sequences $\alpha(\mathbf{X})$ can be expressed by a transformation $\gamma(\mathbf{X})$ from our proposed model, that is, $\gamma(\mathbf{X}) = \alpha(\mathbf{X})$. Note that $\gamma(\mathbf{X})$ only uses four operations, namely, the sequence binary operations $\times$, $\div$, $+$ and $-$. This means that from a theoretical point of view, three operations from the our proposed signal algebra are sufficient to express any sequence transformation. It is important to note that Theorem 1 does not say anything about the algorithmic efficiency of γ nor how can γ be built given α. Moreover, the proof makes use of the concept of sequence polynomial which can be seen as particular type of sequence transformation.

Theorem 1 allows us to state that the proposed model is capable of expressing any transformation from a finite digital signal into a finite digital signal, where the input

digital signal takes values from a finite subset of the real numbers. Therefore, our proposed signal algebra is complete or sufficient in the expression of signal to signal transformations.

Chapter 6
SQL Implementation

In this chapter, we focus on how the proposed set of operations can be implemented in a DBMS. This implementation is very important because it allows us to extend DBMS functionality. It provides to DBMS the ability to perform not only data management but also signal processing using SQL statements. For this purpose, we translate the operations from the proposed model into SQL statements. Moreover, we propose a new SQL clause named `SLIDING`, for the implementation of signal processing sliding windows in RDBMS.

6.1 SQL Implementation

We described in Chapter 3 the methods for storing signals in a DBMS. The data structure used to store a signal depends on the method chosen. The method for storing sequences used in the implementation described in this chapter is the Attribute-Store. That is, we use a relation for storing a signal. The main reason of this choice is that Attribute-Store allows us to take advantage of DBMS in terms of data integrity, security and availability. Moreover, using relations allows us to have consistency between the proposed signal algebra and the implementation, in the sense that, in both cases, sequences are represented as relations.

Most of the sequence operations can be implemented straightforwardly the SQL standard's clauses, allowing us to leverage on the optimizers that query processing engines already have. Some other operations still require the use of User Defined Functions (UDF) for their computation in a RDBMS.

6.1.1 Binary Operations Between Sequences

Since the signal model we propose and the abstract model of RDBMS, are both based on set theory, we can obtain an implementation from the signal model in a straightforward manner. We simply use a relation (a set of pairs) to store a digital signal in a DBMS. This gives us the possibility of using SQL statements to compute signal operations. Let us begin with the SQL statements to compute binary operations between signals.

Query 1 (Binary Operations Between Sequences)**.** *Let us consider the relations* A⟨TimeIndex, Value⟩ *and* B⟨TimeIndex, Value⟩ *representing two signals of the same length. The binary operations expressed in Equation (4.3), denoted here by* `OP`*, can be implemented by the following SQL query.*

```
SELECT A.TimeIndex,A.Value OP B.Value AS Result
FROM A,B
WHERE A.TimeIndex=B.TimeIndex
ORDER BY A.TimeIndex;
```

The previous SQL statement works for `OP` $\in \{+, -, \times, \div\}$. Please note that we can also use the `NATURAL JOIN` clause for a more efficient execution. Nevertheless, for the purpose of this dissertation, we decided to use cross join for a clearer description, as in the previous query. For the case of maximum (max) and minimum (min) operations, we can use the `CASE` expression. For example, to determine the maximum values between elements of the two signals at each index position, we use the following SQL query.

Query 2 (Binary max/min Operations Between Sequences)**.** *Let us consider the relations* A⟨TimeIndex, Value⟩ *and* B⟨TimeIndex, Value⟩ *representing two signals of*

the same length. The binary max *operation can be implemented by the following SQL query.*

```
SELECT A.TimeIndex,
   CASE
      WHEN A.Value >= B.Value THEN A.Value
          ELSE B.Value
   END AS Result
FROM A,B
WHERE A.TimeIndex=B.TimeIndex
ORDER BY A.TimeIndex;
```

The query result is a relation representing a signal whose elements are the maximum values between the signals A and B at each index position. In the case of the minimum, the relational operator ">=" must be replaced by "<=".

Note that in the previous queries, the relations A and B store a single signal each. When the relations store a set of signals instead of a single signal, the queries must be modified accordingly. The following query is similar to Query 1 but instead of single signals now the relations store a set of signals of the same length. In case of implementing binary operations between signals of different length, we can homogenize the lengths of the signals by applying a padding or trimming process.

Query 3. *[Binary Operations Between Sets of Sequences] Let us consider the relations* A⟨signaId, TimeIndex, Value⟩ *and* B⟨signaId, TimeIndex, Value⟩ *representing two set of signals of the same length.The binary operations, denoted by* `OP`*, can be implemented for two set of signals by the following SQL query.*

```
SELECT A.signaId, B.signaId, A.TimeIndex,A.Value OP B.Value AS Result
FROM A,B
```

```
WHERE A.TimeIndex=B.TimeIndex
ORDER BY  A.signaId, B.signaId,A.TimeIndex,;
```

Query 3 implements the binary operation `OP` between two sets of signals. Each signal of the relation A is paired with every signal in relation B. The output of the query will be all the resulting sequences of the binary operation between every combination of signals, computed at every time position. In general, we can implement any binary operation between sets of signals following a similar approach to the previous query.

6.1.2 Operations Between Sequences and Scalars

Similar to the previous query, the implementation of operations between sequences and scalars can be expressed using SQL code.

Query 4 (Binary Operations Between Sequences and Scalars)**.** *Let us consider the relation* A⟨TimeIndex, Value⟩ *representing a signal and a scalar constant denoted by* `S`*. The SQL query for the operations* `OPS` $\in \{+, -, \times, \div\}$ *which are expressed by Equation (4.9) and Equation (4.10) can be implemented as follows.*

```
SELECT A.TimeIndex,A.Value OPS S AS Result
FROM A
ORDER BY A.TimeIndex;
```

For the operations max/min between sequences and scalars, we can use a similar approach as used in Query 2. In the case of exponentiation (^), the function `POWER` is available as part of the SQL:2003 standard [17].

Query 5 (Exponentiation)**.** *Let us consider the relation* A⟨TimeIndex, Value⟩ *representing a signal and a scalar constant denoted by* `S`*. The* `POWER` *function can be used to implement exponentiation as follows.*

```
SELECT A.TimeIndex,POWER(A.Value,S) AS Result
FROM A
ORDER BY A.TimeIndex;
```

The operation $\log_k$ can be obtained by using $\log_k(a) = \frac{\log_e(a)}{\log_e(k)}$ which can be implemented using the function `LN` from the SQL:2003 standard.

6.1.3 Unary Functions over Sequences

The list of numeric (scalar) functions in the current SQL standard is limited in quantity. This kind of functions can be used to compute Unary Functions over Sequences from the signal model. The functions `ABS`, `MOD`, `CEIL`, `FLOOR`, and `SQRT` are some of the functions included in such list. For instance, the computation of the absolute value of each element value of a signal A can be obtained using the following query.

Query 6 (Absolute Value)**.** *Let us consider the relation* A⟨TimeIndex, Value⟩ *representing a signal. The absolute value of each element of the signal stored in the relation A can be implemented as follows.*

```
SELECT A.TimeIndex,ABS(A.Value) AS Result
FROM A
ORDER BY A.TimeIndex;
```

Other types of functions, such as trigonometric functions, are not part of the current SQL standard [18]. Nevertheless, if a given numeric function is not defined

in a DBMS, since the standard SQL:1999 [16] it is possible to utilize a User Defined Function (UDF), allowing users to create their own functions. Therefore, any unary function over sequences can be created following the SQL standard by means of an implementation using UDFs.

A UDF is basically a program that is executed on a DBMS. It performs some operation based on an input and returns a single or multiple results [7]. This result is typically a tuple or a relation. The UDFs can be written in many different programming languages and they use any kind of control structures that the language provides as well as any SQL operation that is available in the DBMS. This flexibility allows UDFs to perform any type of task, even tasks that are not directly related to SQL. UDFs once compiled can be executed in any "SELECT" statement. There are two main classes of UDFs, *scalar* and *aggregate*. Scalar UDFs take one or more parameters values and return a single value, producing one value for each input tuple. Aggregate UDFs return one row for each distinct grouping of column value combinations and a column with some aggregation [64].

6.1.4 Logic Operation over Sequences

A logic operation over a sequence can be implemented with a `CASE` expression which is part of the SQL standard since SQL-92 [15]. As an example, let us consider a situation where we are interested in finding the position of each value in a signal A that is greater than a scalar K. We can figure out this and obtain a result signal with 1 at each position where this condition satisfied and 0 elsewhere by using the following SQL statement.

Query 7 (Logic Operation)**.** *Let us consider the relation* A⟨TimeIndex, Value⟩ *representing a signal. The elements of the signal greater than a scalar* K *can be found*

implementing the following query.

```
SELECT A.TimeIndex,
        CASE
                WHEN A.Value > K THEN 1
                ELSE 0
        END AS Result
FROM A
ORDER BY A.TimeIndex;
```

6.1.5 Aggregate Functions over Sequences

The aggregate functions of maximum (MAX), minimum (MIN) and sum (SUM), can be implemented by the functions `MAX`, `MIN`, and `SUM`, respectively. These functions are included in the SQL-92 standard. Other aggregate functions included in such SQL standard are `AVG` and `COUNT`. Let us denote `AGG` $\in$ {`MAX`, `MIN`,`SUM`, `AVG`, `COUNT`}. The implementation of these operation is shown in the following query.

Query 8 (Aggregate Function)**.** *Let us consider the relation* A⟨TimeIndex, Value⟩ *representing a signal. An aggregate function over a signal can be implemented as follows.*

```
SELECT A.TimeIndex, AGG(A.Value) OVER (PARTITION BY NULL) AS Result
FROM A
ORDER BY A.TimeIndex;
```

Aggregate functions require to be accompanied by the `OVER` clause in order to define groups of tuples based on attributes. In Query 8 only one signal is stored in relation A. In such case, the `OVER (PARTITION BY NULL)` is used. When the

relation stores a set of signals instead of a single signal, the `OVER` clause must specify attributes to define a partition. The following query implements an aggregate function on a relation that stores a set of signals of the same length.

Query 9. *[Aggregate Function on Set of Sequences] Let us consider the relation* B⟨signaId, TimeIndex, Value⟩ *representing a set of signals of the same length where the attribute* signaId *identifies each signal. The aggregate functions, denoted by* `AGG`*, can be implemented by the following SQL query.*

```
SELECT DISTINCT B.signaId, AGG(B.Value)
   OVER (PARTITION BY B.signaId) AS Result
FROM B
ORDER BY B.SignaId;
```

Query 9 implements an aggregate function on relation B representing a set of signals. The output of the query will be all the aggregate results from each signal. This query makes us of the clause `DISTINCT` that allows us to return only the aggregate result of each distinct *signaId*.

In particular, the product (Π) aggregate function described in Section 4.2 is not currently part of the SQL standard. Nevertheless, this aggregate function and other custom made aggregations can be implemented in a DBMS because the SQL standard allows users to create their own aggregate functions by means of User Defined Functions.

6.1.6 Indexing Transformations over Sequences

The process of rearranging the positions of the elements of a sequence can also be done by using a UDF. Nevertheless, simpler rearrangements can be obtained using regular

SQL statements, like the reflection transformation mentioned in Definition 4.27. The following query shows the implementation of a reflection rearrangement.

Query 10 (Reflection)**.** *Let us consider the relation* A⟨TimeIndex, Value⟩ *representing a signal. A reflection rearrangement of the signal can be obtained implementing the following query.*

```
SELECT -1 + ROW_NUMBER() OVER (ORDER BY A.TimeIndex DESC),
        A.Value AS Result
FROM A;
```

Here, we used the `ROW_NUMBER() OVER()` clause. This clause allows us to display the row number of the resultant relation. The operation -1 + `ROW_NUMBER()` is needed to adjust the numbering to start from 0. If the relation A stores more than one signal, with a schema $A\langle signaId,\ TimeIndex,\ Value\rangle$, the `OVER` clause must be modified accordingly and obtain all signals reflected. That is,

```
OVER (PARTITION BY signaID ORDER BY A.TimeIndex DESC).
```

More sophisticated rearrangements can be accomplished using a UDF. For instance, let us define a UDF named *Rearrangement(input_signal)* which is described using Algorithm 1.

Algorithm 1: Indexing Operation Algorithm

Input: Sig Input Signal
Output: Transformed Sequence $Nseq$

$pos \leftarrow 0$;
while $pos < Length(Sig)$ **do**
  $Newposition \leftarrow Rearrangement(pos)$;
  $Nseq[Newposition] \leftarrow Sig[pos]$;
  $pos \leftarrow pos + 1$;
end while
return $Nseq$;

Algorithm 1 works as follows. The algorithm takes a signal (Sig) as input. Then, an indexing transformation is applied at each position using $Rearrangement(pos)$. The new index is assigned to $Newposition$. Afterwards, the value of $Newposition$ is used to assign the element $Sig[pos]$) to $Nseq[Newposition]$. The output is the new transformed sequence $Nseq$. The user defined function implementing Algorithm 1, using signal A, can be used in an SQL query with the following syntax.

Query 11 (Rearrangement using a Function)**.** *Let us consider the relation* A *with a schema* A⟨TimeIndex, Value⟩ *representing a signal. A rearrangement of the signal can be obtained implementing the UDF* Rearrangement *using following query.*

```
SELECT Rearrangement(A.TimeIndex) AS NewTimeIndex, A.Value
FROM A
ORDER BY NewTimeIndex;
```

The function *Rearrangement* transforms each index position into a new position. Therefore, the tuples obtained from the query consist of the new position and the value associated to it. Finally, the `ORDER BY` arranges the following tuples using the new positions.

Let consider the case where the old and new positions are stored in a relation. For instance, a relation with a schema $R\langle TimeIndex,\ NewTimeIndex\rangle$. In such case, a query to rearrange the index position of a signal stored in a relation A according to the relation R can be implemented as follows.

Query 12 (Rearrangement using a Relation)**.** *Let us consider the relation* A *with a schema* A⟨TimeIndex, Value⟩ *representing a signal and a relation* R *with a schema* R⟨TimeIndex, NewTimeIndex⟩ *that establishes an indexing transformation. A rearrangement of the signal using the relation R can be obtained implementing the following query.*

```
SELECT R.NewTimeIndex, A.Value
FROM R,A
WHERE A.TimeIndex=R.TimeIndex
ORDER BY R.NewTimeIndex;
```

The relation R can be thought of as an equivalent representation of the function *Rearrangement*. The transformation is here accomplished by a cross join between relations R and A.

6.1.7 Sequence Processing Operations

The task of implementing a processing operation in a DBMS requires a User Defined Function (UDF) and an array structure in addition to the stored input signal. That is, one of the sequences, the pattern sequence, must be stored using the Tuple-Store method. The reason for using the Tuple-Store instead of the Attribute-Store for storing the pattern sequence is due to a limitation of the DBMS' on the use of the `OVER` clause. Specifically, the `OVER` clause can only be applied on attributes that share the same windowing definition specified by the clause. It is not possible to apply the `OVER` clause to two or more attributes when these require different window definitions. This limitation impedes to join groups or windows of tuples from two different attributes and implement a function using the groups of both attributes. In order to avoid this limitation we need to store one of the sequences, in this case the pattern sequence, as an array data type. This allows us to apply an operation between the array sequence and the input sequence stored as a relation. In the rest of this section, we describe the implementation of a processing operation by a UDF, using an array structure to store the pattern sequence and a relation to store a input sequence.

Let us consider the relations $A\langle TimeIndex,\ Value\rangle$ and $P\langle Sequence\rangle$ where the relation A represents the stored digital signal and the relation P is used to store a pattern sequence using an array data type. We define a UDF named *Processing(input_signal, pattern_sequence)*, which is a general function to implement a processing operation on stored signals. Its arguments are the input signal and the pattern sequence. Basically, the UDF *Processing* performs a binary operation between the sequences followed by an aggregate function. In order to perform the processing operation, we make use of the `OVER` clause. The `OVER` clause allows us to define an sliding window and is part of the standard since SQL:1999. The arguments of the `OVER` clause must be modified accordingly to a specific processing operation. In this particular case, at each row, a window is created using the option `ROWS BETWEEN` from the `CURRENT ROW` including the `FOLLOWING Length(P.sequence)` rows. The value `Length(P.sequence)` is the length of the pattern sequence stored in attribute *P.sequence* and is used to specify the size of the sliding window. Naturally it was necessary to provide an ordinal structure to the window, this is done by the option `ORDER BY A.TimeIndex` of the `OVER` clause. This procedure is described by Algorithm 2.

Algorithm 2: Processing Operation Algorithm

Input: Sig Input Signal,
$Pseq$ Pattern Sequence
Output: Output Element of a Sequence

1 $Seq \leftarrow Define\_Sequence(Sig)$;
2 $Bin\_oper\_out\_Seq \leftarrow OP(Seq, Pseq)$;
3 $Aggregate\_oper\_out \leftarrow AGG(bin\_oper\_out\_Seq)$;
4 *return* $Aggregate\_oper\_out$;

Algorithm 2 works as follows. First, a subsequence (Seq) of a signal (Sig) is defined (by means of an SQL query which is described below). Afterwards, a binary operation of sequences `OP` is performed between Seq and the Pattern sequence ($Pseq$).

Then, the resulting sequence (*Bin_oper_out_Seq*), is reduced to a scalar value by means of an aggregate function. The scalar result value (*Aggregate_oper_out*) is returned as the output of the function. An abstract view of a similar type of algorithm is the linear filtering whose expression is the following equation.

$$\mathbf{X} \otimes \mathscr{H} = \{(j, w[j]) : w[j] = \sum(\mathbf{X} \times \mathbf{H}_j),\ j \in \mathbb{Z}_N\}. \tag{6.1}$$

where $\mathbf{X}$ represents the input signal and the pattern is abstracted by the kernel structure $\mathscr{H}$. The following query shows the implementation of Equation (6.1).

Query 13. *[Processing Operation] Let us consider the relations* A⟨TimeIndex, Value⟩ *representing a signal, and the relation* P⟨Sequence⟩ *representing a pattern sequence stored as an array data type. A User Defined Function (*Processing*) implementing Algorithm 2 can be used in a DBMS using an SQL statement with the following syntax.*

```
SELECT A.TimeIndex,Processing(A.Value,P.Sequence)
        OVER (ORDER BY A.TimeIndex
        ROWS BETWEEN CURRENT ROW AND Length(P.sequence) FOLLOWING)
        AS Result
FROM A,P;
```

Query 13 assumes that just one signal is stored in the relation A and that just one pattern is stored in relation P. Nonetheless, it is possible expand the SQL statement for sets of sequences and sets of patterns.

6.2 SLIDING: An SQL Clause Proposal

As described in the previous section, current RDBMS' supporting the SQL standard require mixing different storing methods for the implementation of processing operations between sequences. This is necessary due to the nature of the `OVER` clause

operation. Particularly, the `OVER` clause is limited, given that it can only be applied on attributes passed to functions as arguments that share the grouping or windowing definition specified by the clause. That is, the same grouping definition must be applied to the attributes. For instance, in analytic functions that require two or more attributes as arguments, only one window definition can be specified by the `OVER` clause. Moreover, when different windows definitions are required by the attributes of analytic functions, the current SQL standard does not allow the use of subqueries as arguments on those functions. In order to deal with these restrictions, it was necessary to use an array structure to store one of the sequences involved in the operation. In turn, this implies that the signal stored as an array must fit in main memory. This is a limitation of the current SQL standard regarding its application in processing signal data. We propose the addition of a new SQL clause that will allow us to deal with these restrictions without the necessity of mixing different storing methods for signal data. We consider that this new SQL clause can improve to a great extent the applicability of RDBMS for management and processing signal data.

A common characteristic of sequence processing operations is that all of them are based on an sliding window. In particular, a processing operation requires the definition of an sliding window on the input sequence. Afterwards, an operation is applied between each defined window and a whole pattern sequence. However, due to the limitation of the current SQL standard, we cannot implement an sliding window using two sequences stored as relations. In order to avoid using different data types to store sequences, we propose the addition of a new SQL clause. This clause, `SLIDING`, will allow us to use only one storing method (Attribute-Store) for signal data in a DBMS. In order to compute a processing operation, an sliding window must be defined on an input sequence stored as a relation. This can be done using the regular `OVER` clause. Moreover, each window created by the `OVER` clause must interact with a

pattern sequence, also stored as a relation. In this step is where the new SQL clause is required.

Let us illustrate the semantics of the `SLIDING` clause with an example, as shown in Figure 6.1. There is a relation L representing the input sequence (a, b, c) with a schema $L\langle T,\ V\rangle$. The pattern sequence (x, y) is represented by relation M with a schema $M\langle T,\ V\rangle$. These relations are depicted in Figure 6.1(a). We start the description of this example by showing its full implementation on a DBMS, using the proposed `SLIDING` clause.

Query 14 (Function Implementation using the SLIDING clause)**.** *Let us consider the relations* L⟨T, V⟩ *and* M⟨T, V⟩ *representing a pair of sequences. An analytic* Function *accepts two attributes as arguments. An sliding window using the two attributes can be obtained by the following SQL statement.*

```
SELECT L.T, Function(L.V, (SELECT M.V FROM M ORDER BY M.T))
SLIDING OVER (ORDER BY L.T ROWS BETWEEN CURRENT ROW 1 PRECEDING)
FROM L
ORDER BY L.T;
```

In order to define an sliding window on relation L, the `OVER` clause is used. In this case, we use the SQL expression

`OVER (ORDER BY L.T ROWS BETWEEN CURRENT ROW 1 PRECEDING).`

Here, an sliding window of length 2 is defined. The two rows included in each window is the current row and the previous row.

In Figure 6.1(b), we describe window partitioning on the input sequence stepwise. In step 1, there is not a previous row, in such case, a `NULL` value is appended in the first window. In step 2, the sliding window moves forward one row defining the

subsequence (a, b). In step 3, the current row is the tuple $\langle 2,\ \mathrm{c}\rangle$ and the window defines the subsequence (b, c). Now, it is necessary to define the sequence represented by the relation M, as shown in Figure 6.1(d). In the case of processing operations, the pattern sequence, here represented by relation M, does not require the definition of window subsequences. Instead, the complete sequence must be defined as a whole. The definition of the complete sequence is equivalent to the relation obtained from the following SQL statement and must be executed as a subquery.

Query 15 (Complete Sequece Definition as part of an Sliding Window)**.** *Let us consider the relations* M⟨T, V⟩ *representing a sequence. In order to define a sequence as a whole in an sliding window, we use an SQL statement with the following syntax.*

```
SELECT V
FROM M
ORDER BY T;
```

The corresponding window-defined subsequences obtained from the sliding window are depicted in Figure 6.1(c). In the SQL statement, the inclusion of Query 15 as an argument on the function should have make the sliding window work. That is,

```
Function(L.V, (SELECT V FROM M ORDER BY T))
OVER (ORDER BY L.T ROWS BETWEEN CURRENT ROW 1 PRECEDING)
```

Unfortunately, including a subquery as an argument on analytic functions is not allowed by the current SQL standard. We propose to remove that restriction by using the `SLIDING` clause, that is,

```
Function(L.V, (SELECT V FROM M ORDER BY T))
SLIDING OVER (ORDER BY L.T ROWS BETWEEN CURRENT ROW 1 PRECEDING)
```

The `SLIDING` clause, allows us here to include a subquery as an argument of the analytic function. Note that the `OVER` clause is defining an sliding window on relation L. The corresponding window-defined subsequences obtained from the `OVER` clause and the pattern sequence generated by the subquery, become the arguments of the analytic function as shown in Figure 6.2.

The behavior of the analytic function associated to the `OVER` clause is modified by placing the `SLIDING` clause before the `OVER` clause, that is,

```
SLIDING
OVER
( [ <PARTITION BY clause> ]
[ <ORDER BY clause> ]
[ <ROW or RANGE clause> ])
```

The combination `SLIDING OVER()` allows the usage of subqueries as arguments of an analytic function giving the capability of constructing an sliding window operation. The operation of the `OVER` clause remains unaltered with respect to its definition on the SQL:1999 standard.

In the previous section, Query 13 implements a UDF named *Processing*, built from Algorithm 2, using an array to store the P sequence. Now, by using the `SLIDING` clause, we can now use only relations to store both signals. Moreover, we are able to implement the same function using a subquery as argument of the user-defined analytic function *Processing*. The following query implements the UDF *Processing* using the `SLIDING` clause.

Query 16 (Processing Operation using the SLIDING Clause)**.** *Let us consider the relations* A⟨TimeIndex, Value⟩ *representing a signal, and the relation* P⟨TimeIndex, Value⟩ *representing a pattern. A User Defined Function* Processing *implementing*

Algorithm 2 can be used in a DBMS using an SQL statement with the following syntax.

```
SELECT A.TimeIndex,
        Processing(A.Value,(SELECT P.Value
        FROM P ORDER BY P.TimeIndex))
        SLIDING OVER (ORDER BY A.TimeIndex
        ROWS BETWEEN CURRENT ROW AND (SELECT COUNT(*)
        FROM P) FOLLOWING )
FROM A
ORDER BY A.TimeIndex;
```

In this example, the window defined by the `OVER` clause considers ahead rows from the current row. It uses the subquery `SELECT COUNT(*) FROM P` to specify the number of following rows to be considered. Different arrangements of the sliding window and its relative position to the current row can be defined using the options included in the `OVER` clause as usual.

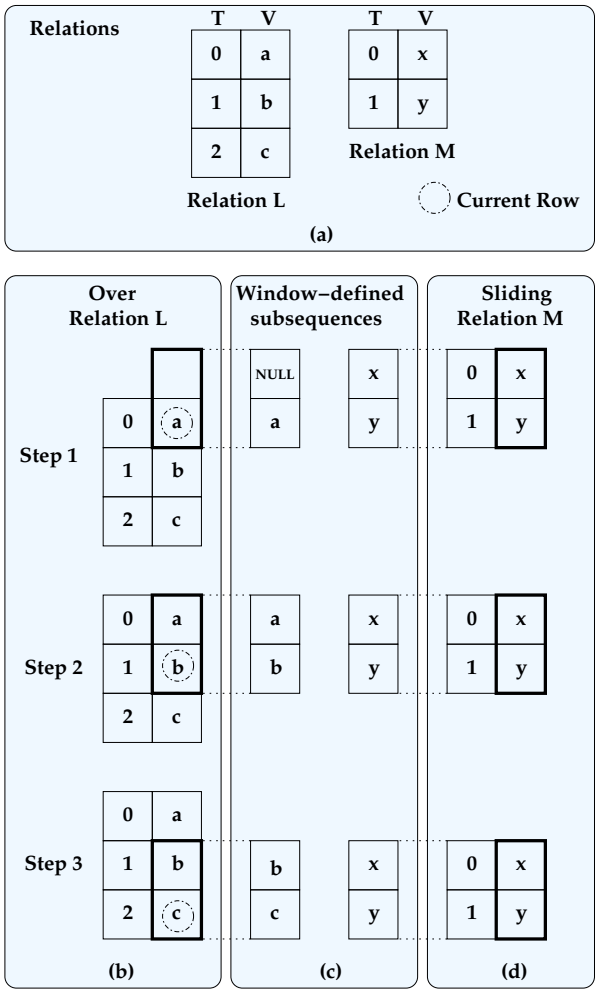

Figure 6.1: An illustration of the semantics of the SLIDING clause. (a) Relations L and M, (b) Window partitioning on an input sequence, (c) Sequence to sequence operation, and (d) Sliding of the pattern over the window defined on the input sequence.

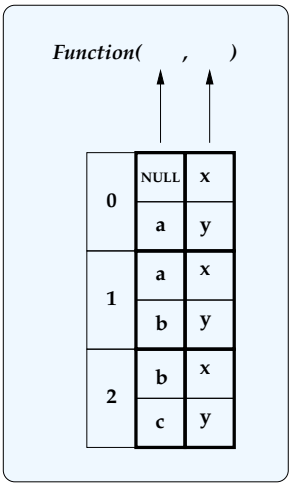

Figure 6.2: Analytic function with a pair of window-defined arguments.

Chapter 7
Conclusions

In this dissertation, we present a formal data model for the representation of signal data and the expression of signal analysis algorithms. The main motivation behind this research project was the need to define a model for digitally storing, managing and processing ECG signal data. Given this motivation, we considered ECGs as finite digital signals. Sequences, represented as set of pairs, are used as the abstract objects to represent digital signals. The proposed model provides a set of operations for the manipulation of sequences. Both, the formal representation of sequences and the set of operations can be easily integrated with current relational database systems since both are based on set theory. We advocate for abstracting digital signals as relations (set of pairs). Traditionally, signal processing and signal data management have been addressed separately in different computing platforms. Such an approach is adequate for small scale data sets but highly inefficient for large scale data. For instance, in a massive data scenario, processing tasks using the file-system and ad-hoc approaches cannot be implemented without a complex data management due to the discrepancy between data size and memory availability. Moreover, data integrity can be threaten by transferring data from one computing platform to another.

The main contribution of this dissertation is that it provides the capability of merging signal processing and signal data management into a single data model. This model provides a clear mathematical notation for signal processing algorithms in database management systems. This facilitates the process of translating mathematical equations into lines of code of a programming language. Moreover, its conciseness

enables the straightforward application of optimization techniques for processing algorithms implemented in database systems. Our proposed model lays the foundations for large-scale signal data systems, where the tasks of signal data processing and data management can be accomplished in the same database environment. Furthermore, we provide evidence that this formal model can easily capture common ECG processing and querying tasks needed for ECG data analysis, such as, noise filtering, feature extraction and similarity search. Moreover, we describe how the model can be implemented in a DBMS by translating the set operations into SQL statements and User Defined Functions (UDF). Additionally, we propose a new SQL clause named `SLIDING`. We argue that the addition of this new clause to the SQL standard can improve to a great extent the applicability of RDBMS for management and processing signal data.

Given the demand of database systems capable of managing and analyzing very large volumes of signal data, we intend to develop the concrete counterpart of the proposed model. We plan to build an integral system for signal management, processing, and analysis using RDBMS. The operations of the system will be founded in the signal algebra proposed in this dissertation. We would like to compare the execution efficiency of the operations implemented in a DBMS against an ad-hoc implementation using the file-system. Moreover, there are some opportunities to improve the current model. For instance, it will be of interest to consider multidimensional signals and investigate to what extent the proposed model in relevant for such types of signals. Furthermore, we would like to extend the current model to cover the expression of processing techniques computed via recursion.

Bibliography

[1] B. Abibullaev and H.-D. Seo. A New QRS Detection Method Using Wavelets and Artificial Neural Networks. *Journal of Medical Systems*, 35(4):683–691, 2011.

[2] V.X. Afonso, Willis J. Tompkins, T.Q. Nguyen, and Shen Luo. ECG Beat Detection using Filter Banks. *IEEE Transactions on Biomedical Engineering*, 46(2):192–202, 1999.

[3] P. M. Agante and J. P. Marques de Sa. ECG Noise Filtering Using Wavelets with Soft-Rhresholding Methods. In *Proceedings of Computers in Cardiology*, pages 535–538, Sep 1999.

[4] J.A. Van Alste and T.S. Schilder. Removal of Base-Line Wander and Power-Line Interference from the ECG by an Efficient FIR Filter with a Reduced Number of Taps. *IEEE Transactions on Biomedical Engineering*, 32(12):1052–1060, 1985.

[5] F. Badilini, A. J. Moss, and E. L. Titlebaum. Cubic Spline Baseline Estimation in Ambulatory ECG Recordings for the Measurement of ST Segment Displacements. In *Proceedings of Engineering in Medicine and Biology Society*, pages 584–585, Oct 1991.

[6] S. Barro, M. Fernandez-Delgado, J. A Vila-Sobrino, C.V. Regueiro, and E. Sanchez. Classifying multichannel ecg patterns with an adaptive neural network. *IEEE Engineering in Medicine and Biology Magazine*, 17(1):45–55, 1998.

[7] H. Bedoya, F. Cruz, and D. Lema. *Stored Procedures, Triggers and User Defined Functions on DB2 Universal Database for ISeries*. IBM redbooks. IBM, International Technical Support Organization, 2004.

[8] V. Bhateja, S. Urooj, R. Verma, and R. Mehrotra. A Novel Approach for Suppression of Powerline Interference and Impulse Noise in ECG Signals. In *Proceedings of the International Conference on Multimedia, Signal Processing and Communication Technologies*, pages 103–107, Nov 2013.

[9] I. I. Christov, I. A. Dotsinsky, and I. K. Daskalov. High-Pass Filtering of ECG Signals using QRS Elimination. *Medical and Biological Engineering and Computing*, 30(2):253–256, 1992.

[10] C.-H. H. Chu and E. J. Delp. Impulsive Noise Suppression and Background Normalization of Electrocardiogram Signals using Morphological Operators. *IEEE Transactions on Biomedical Engineering*, 36(2):262–273, 1989.

[11] G.D. Clifford, F. Azuaje, and P.E. McSharry. *Advanced Methods and Tools for ECG Data Analysis*. Artech House Engineering in Medicine & Biology series. Artech House, 2006.

[12] L.A. D'Alotto, C.R. Giardina, and H. Luo. *A Unified Signal Algebra Approach to Two-Dimensional Parallel Digital Signal Processing*. Marcel Dekker Inc, 1998.

[13] J. P. Marques de Sa. Digital FIR Filtering for Removal of ECG Baseline Wander. *Journal of Clinical Engineering*, 8(3):235–240, 1982.

[14] S. Dobbs, N. Schmitt, and H. Ozemek. QRS Detection by Template Matching using Real-Time Correlation on a Microcomputer. *Journal of Clinical Engineering*, 9(3):197–212, 1984.

[15] Database Language SQL. Standard, International Organization for Standardization (ISO), Document ISO/IEC 9075:1992.

[16] Database Language SQL. Standard, International Organization for Standardization (ISO), Document ISO/IEC 9075:1999.

[17] Database Language SQL. Standard, International Organization for Standardization (ISO), Document ISO/IEC 9075:2003.

[18] Database Language SQL. Standard, International Organization for Standardization (ISO), Document ISO/IEC 9075:2011.

[19] W. Dreyer, A. K. Dittrich, and D. Schmidt. Research Perspectives for Time Series Management Systems. *ACM SIGMOD Record*, 23(1):10–15, 1994.

[20] E. A. Emerson. Handbook of Theoretical Computer Science (Vol. b). chapter Temporal and Modal Logic, pages 995–1072. MIT Press, Cambridge, MA, USA, 1990.

[21] O. Ersoy. Real Discrete Fourier Transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 33(4):880–882, 1985.

[22] Z. Faget, P. Rigaux, D. Gross-Amblard, and V. Thion-Goasdoué. Modeling Synchronized Time Series. In *Proceedings of the Fourteenth International Database Engineering; Applications Symposium*, IDEAS '10, pages 82–89. ACM, Aug 2010.

[23] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence Matching in Time-Series Databases. *SIGMOD Record*, 23(2):419–429, 1994.

[24] L. J. Ferrer-Arnau, R. Reig-Bolaño, P. Marti-Puig, A. Manjabacas, and V. Parisi-Baradad. Efficient Cubic Spline Interpolation Implemented with FIR Filters. *International Journal of Computer Information Systems and Industrial Management Applications*, 5(1):098–105, 2013.

[25] T. C. Fu. A Review on Time Series Data Mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[26] A. Gacek and W. Pedrycz. *ECG Signal Processing, Classification and Interpretation: A Comprehensive Framework of Computational Intelligence*. Springer, 1st edition, 2011.

[27] S. K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM Transactions of Database Systems*, 13(4):418–448, 1988.

[28] F. Grandi and F. Mandreoli. A Formal Model for Temporal Schema Versioning in Object-oriented Databases. *Data & Knowledge Engineering*, 46(2):123–167, 2003.

[29] H. Gregersen and C. S. Jensen. Temporal Entity-Relationship Models-a Survey. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):464–497, 1999.

[30] M. Holschneider, R. Kronland-Martinet, J. Morlet, and Ph. Tchamitchian. A Real-Time Algorithm for Signal Analysis with the Help of the Wavelet Transform. In Jean-Michel Combes, Alexander Grossmann, and Philippe Tchamitchian, editors, *Wavelets*, Inverse Problems and Theoretical Imaging, pages 286–297. Springer Berlin Heidelberg, 1990.

[31] W. P. Holsinger, K. M. Kempner, and M. H. Miller. A QRS Preprocessor Based on Digital Differentiation. *IEEE Transactions on Biomedical Engineering*, BME-18(3):212–217, 1971.

[32] R. Jane, P. Laguna, N.V. Thakor, and P. Caminal. Adaptive Baseline Wander Removal in the ECG: Comparative Analysis with Cubic Spline Technique. In *Proceedings of Computers in Cardiology*, pages 143–146, Oct 1992.

[33] I. Jekova, G. Bortolan, and I. Christov. Assessment and Comparison of Different Methods for Heartbeat Classification. *Medical Engineering & Physics*, 30(2):248–257, 2008.

[34] S. Kadambe, R. Murray, and G.F. Boudreaux-Bartels. Wavelet Transform-Based QRS Complex Detector. *IEEE Transactions on Biomedical Engineering*, 46(7):838–848, 1999.

[35] S. T. Karris. *Signals and Systems: with MATLAB Computing and Simulink Modeling*. Orchard Publications, 2008.

[36] E. Keogh, J. Lin, A. W. Fu, and H. VanHerle. Finding Unusual Medical Time-Series Subsequences: Algorithms and Applications. *IEEE Transactions on Information Technology in Biomedicine*, 10(3):429–439, 2006.

[37] M.-S. Kim, Y.-C. Cho, S.-T. Seo, C.-S. Son, and Y.-N. Kim. A New Method of ECG Feature Detection Based on Combined Wavelet Transform for U-Health Service. *Biomedical Engineering Letters*, 1(2):108–115, 2011.

[38] A. S. M. Koeleman, H. H. Ros, and T. J. Van den Akker. Beat-to-Beat Interval Measurement in the Electrocardiogram. *Medical and Biological Engineering and Computing*, 23(3):213–219, 1985.

[39] B.-U. Kohler, C. Hennig, and R. Orglmeister. The Principles of Software QRS Detection. *Engineering in Medicine and Biology Magazine, IEEE*, 21(1):42–57, 2002.

[40] T.I. Laakso and V. Valimaki. Energy-Based Effective Length of the Impulse Response of a Recursive Filter. *IEEE Transactions on Instrumentation and Measurement*, 48(1):7–17, 1999.

[41] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo. Clustering ECG Complexes using Hermite Functions and Self-Organizing Maps. *IEEE Transactions on Biomedical Engineering*, 47(7):838–848, 2000.

[42] P. Laguna, R. Jané, and P. Caminal. Automatic Detection of Wave Boundaries in Multilead ECG Signals: Validation with the CSE Database. *Computers and Biomedical Research*, 27(1):45–60, 1994.

[43] C. Levkov, G. Michov, R. Ivanov, and I. Daskalov. Subtraction of 50 Hz Interference from the Electrocardiogram. *Medical and Biological Engineering and Computing*, 22(4):371–373, 1984.

[44] C. Levkov, G. Mihov, R. Ivanov, I. Daskalov, I. Christov, and I. Dotsinsky. Removal of Power-Line Interference from the ECG: A Review of the Subtraction Procedure. *BioMedical Engineering OnLine*, 4(1):50, 2005.

[45] Y. Li, H. Yan, F. Hong, and J. Song. A New Approach of QRS Complex Detection Based on Matched Filtering and Triangle Character Analysis. *Australasian Physical & Engineering Sciences in Medicine*, 35(3):341–356, 2012.

[46] X. Lian and L. Chen. Subspace Similarity Search under L_p-Norm. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):365–382, 2012.

[47] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11, September 2003.

[48] D. Lopez-Barron, I. Vega-Lopez, and O. Cuen-Tellez. Supporting Query by Content on ECG Data with User Defined Functions. In *Proceedings of the IASTED International Conference Advances in Computer Science*, pages 392–399, April 2013.

[49] S. Mallat. Zero-Crossings of A Wavelet Transform. *Information Theory, IEEE Transactions on*, 37(4):1019 –1033, 1991.

[50] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition, 2008.

[51] S. Mallat and W.-L. Hwang. Singularity Detection and Processing with Wavelets. *IEEE Transactions on Information Theory*, 38(2):617–643, 1992.

[52] S. M. M. Martens, M. Mischi, S. G. Oei, and J. W. M. Bergmans. An Improved Adaptive Power Line Interference Canceller for Electrocardiography. *IEEE Transactions on Biomedical Engineering*, 53(11):2220–2231, 2006.

[53] J. P. Martinez, R. Almeida, S. Olmos, A. P. Rocha, and P. Laguna. A Wavelet-Based ECG Delineator: Evaluation on Standard Databases. *IEEE Transactions on Biomedical Engineering*, 51(4):570–581, 2004.

[54] G. Matheron. *Random Sets and Integral Geometry*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1st edition, 1974.

[55] L. E. McKenzie, Jr. and R. T. Snodgrass. Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. *ACM Computing Surveys*, 23(4):501–543, 1991.

[56] S. H. Meij, P. Klootwijk, J. Arends, and J. R. T. C. Roelandt. An Algorithm for Automatic Beat-to-Beat Measurement of the QT-Interval. In *Proceeding of Computers in Cardiology*, pages 597–600, Sep 1994.

[57] C. R. Meyer and H. N. Keiser. Electrocardiogram Baseline Noise Estimation and Removal using Cubic Splines and State-Space Computation Techniques. *Computers and Biomedical Research*, 10(5):459 – 470, 1977.

[58] K. Minami, H. Nakajima, and T. Toyoshima. Real-time Discrimination of Ventricular Tachyarrhythmia with Fourier-Transform Neural Network. *IEEE Transactions on Biomedical Engineering*, 46(2):179–185, 1999.

[59] S.K. Mitra. *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, 2010.

[60] J. Moraes, M. M. Freitas, F. N. Vilani, and E. V. Costa. A QRS Complex Detection Algorithm using Electrocardiogram Leads. In *Proceedings of Computers in Cardiology*, pages 205–208, Sep 2002.

[61] S.H. Oguz and M.H. Asyali. A Morphology based Algorithm for Baseline Wander Elimination in ECG Records. In *Proceedings of the International Biomedical Engineering Days*, pages 157–160, Aug 1992.

[62] M. Okada. A Digital Filter for the QRS Complex Detection. *IEEE Transactions on Biomedical Engineering*, BME-26(12):700–703, 1979.

[63] A.V. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[64] C. Ordonez. Statistical Model Computation with UDFs. *IEEE Transactions on Knowledge and Data Engineering*, 22(12):1752–1765, 2010.

[65] World Health Organization. *A Prioritized Research Agenda for Prevention and Control of Noncommunicable Diseases*. Nonserial Publication Series. World Health Organization, 2011.

[66] S. Osowski and Tran H.-L. ECG Beat Recognition using Fuzzy Hybrid Neural Network. *IEEE Transactions on Biomedical Engineering*, 48(11):1265–1271, 2001.

[67] J. Pan and W.J. Tompkins. A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3):230–236, 1985.

[68] M. Paredes, D. Rodriguez, and J. Villamizar-Morales. La Estructura Algebraica del Espacio de Señales Unidimensionales. *Revista Integración*, 23(2):15–39, 2005.

[69] M. Y. Park, D. Yoon, N. K. Choi, J. Lee, K. Lee, H. S. Lim, B. J. Park, J. H. Kim, and R. W. Park. Construction of An Open-Access QT Database for Detecting the Proarrhythmia Potential of Marketed Drugs: ECG-ViEW. *Clinical Pharmacology & Therapeutics*, 92(3):393–396, 2012.

[70] S.-C. Pei and C.-C. Tseng. Elimination of AC Interference in Electrocardiogram using IIR Notch Filter with Transient Suppression. *IEEE Transactions on Biomedical Engineering*, 42(11):1128–1132, 1995.

[71] M. Puschel and J.M.F. Moura. Algebraic Signal Processing Theory: Foundation and 1-D Time. *IEEE Transactions on Signal Processing*, 56(8):3572–3585, 2008.

[72] C.M. Rader. DSP History - The Rise and Fall of Recursive Digital Filters. *IEEE Signal Processing Magazine*, 23(6):46–49, 2006.

[73] F. Rincon, J. Recas, N. Khaled, and D. Atienza. Development and Evaluation of Multilead Wavelet-Based ECG Delineation Algorithms for Embedded Wireless Sensor Nodes. *IEEE Transactions on Information Technology in Biomedicine*, 15(6):854–863, 2011.

[74] O. Rioul and M. Vetterli. Wavelets and Signal Processing. *IEEE Signal Processing Magazine*, 8(4):14 –38, 1991.

[75] G. X. Ritter, M. A. Shrader-Frechette, and J. N. Wilson. Image Algebra: A Rigorous and Translucent Way of Expressing All Image Processing Operations. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers, Infrared Image Processing and Enhancement 0781*, pages 116–121, Sep 1987.

[76] G. X Ritter, J. N Wilson, and J. L Davidson. Image Algebra: An Overview. *Computer Vision, Graphics, and Image Processing*, 49(3):297–331, 1990.

[77] A. Ruha, S. Sallinen, and S. Nissila. A Real-Time Microprocessor QRS Detector System with a 1-ms Timing Accuracy for the Measurement of Ambulatory HRV. *IEEE Transactions on Biomedical Engineering*, 44(3):159–167, 1997.

[78] Pon L. S., Sun M., and R. J. Sclabassi. The Bi-Directional Spike Detection in EEG Using Mathematical Morphology and Wavelet Transform. In *Proceedings of the International Conference on Signal Processing*, pages 1512–1515, Aug 2002.

[79] J. S. Sahambi, S. N. Tandon, and R. K. P. Bhatt. Using Wavelet Transforms for ECG Characterization. An On-Line Digital Signal Processing System. *IEEE Engineering in Medicine and Biology Magazine*, 16(1):77–83, 1997.

[80] R. Sears, C. Van-Ingen, and J. Gray. To BLOB or Not To BLOB: Large Object Storage in Database or Filesystem? TechReport MSR-TR-2006-45, Microsoft Research, New York, NY, USA, April 2006.

[81] A. Segev and R. Chandra. A Data Model for Time-Series Analysis. In *Proceedings of Advanced Database Systems*, pages 191–212, Sep 1993.

[82] L. Senhadji, G. Carrault, J. J Bellanger, and G. Passariello. Comparing Wavelet Transforms for Rrecognizing Cardiac Patterns. *IEEE Engineering in Medicine and Biology Magazine*, 14(2):167–173, 1995.

[83] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, Inc., Orlando, FL, USA, 1st edition, 1983.

[84] J. O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, http://www.w3k.org/books/, 2007.

[85] Richard Snodgrass. The Temporal Query Language TQuel. *ACM Transasctions on Database Systems*, 12(2):247–298, 1987.

[86] E. Soria-Olivas, M. Martinez-Sober, J. Calpe-Maravilla, J.F. Guerrero-Martinez, J. Chorro-Gasco, and J. Espi-Lopez. Application of Adaptive Signal Processing for Determining the Limits of P and T Waves in an ECG. *IEEE Transactions on Biomedical Engineering*, 45(8):1077–1080, 1998.

[87] L. Sornmo. Time-Varying Filtering for Removal of Baseline Wander in Exercise ECGs. In *Proceedings of Computers in Cardiology*, pages 145–148, Sep 1991.

[88] L. Sornmo and P. Laguna. *Electrocardiogram (ECG) Signal Processing*. John Wiley & Sons, Inc., 2006.

[89] G. Speranza, G. Nollo, F. Ravelli, and R. Antolini. Beat-to-Beat Measurement and Analysis of the R-T Interval in 24 h ECG Holter Recording. *Medical Biological Engineering Computing*, 31(5):487–49, 1993.

[90] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran, and S. Zdonik. C-Store: A Column-Oriented DBMS. In *Proceedings of the International Conference on Very Large Data Bases*, pages 553–564, Sep 2005.

[91] Y. Sun, K. Chan, and S. Krishnan. Characteristic Wave Detection in ECG Signal using Morphological Transform. *BMC Cardiovascular Disorders*, 5(1):1–7, 2005.

[92] S. Suppappola and Ying Sun. Nonlinear Transforms of ECG Signals for Digital QRS Detection: A Quantitative Analysis. *IEEE Transactions on Biomedical Engineering*, 41(4):397–400, 1994.

[93] L. Sörnmo and P. Laguna. *The Electrocardiogram - A Brief Background*. John Wiley & Sons, Inc., 2006.

[94] N. V. Thakor and Zhu Y.-S. Applications of Aadaptive Filtering to ECG Analysis: Noise Cancellation and Arrhythmia Detection. *IEEE Transactions on Biomedical Engineering*, 38(8):785–794, 1991.

[95] P. E. Trahanias. An Approach to QRS Complex Detection using Mathematical Morphology. *IEEE Transactions on Biomedical Engineering*, 40(2):201–205, 1993.

[96] M. Unser. Splines: A Perfect Fit for Signal/Image Processing. *IEEE Signal Processing Magazine*, 16(6):22–38, 1999.

[97] M. Unser, A Aldroubi, and M. Eden. B-Spline Signal Processing. II. Efficiency Design and Applications. *IEEE Transactions on Signal Processing*, 41(2):834–848, 1993.

[98] S. Yan, K. Chan, and S. Krishnan. ECG Signal Conditioning by Morphological Filtering. *Computers in Biology and Medicine*, 32(6):465–479, 2002.

[99] F. Zhang and Y. Lian. QRS Detection Based on Multiscale Mathematical Morphology for Wearable ECG Devices in Body Area Networks. *IEEE Transactions on Biomedical Circuits and Systems*, 3(4):220–228, 2009.

[100] F. Zhang and Y. Lian. QRS Detection Based on Morphological Filter and Energy Envelope for Applications in Body Sensor Networks. *Journal of Signal Processing Systems*, 64(2):187–194, 2011.