Formatting and Searching a Massive, Multi-parameter Clinical Information Database

by

Tin Htet Kyaw

S.B., MIT, Electrical Engineering and Computer Science (2004)

and

S.B., MIT, Management Science (2005)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author ....
Department of Electrical Engineering and Computer Science
August 18, 2005

Certified by ....
Professor Roger G. Mark
Distinguished Professor in Health Sciences and Technology
Professor of Electrical Engineering
Thesis Supervisor

Accepted by
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Formatting and Searching a Massive, Multi-parameter Clinical Information Database

by

Tin Htet Kyaw

Submitted to the Department of Electrical Engineering and Computer Science
on August 18, 2005, in partial fulfillment of the
requirements for the degrees of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Formatting data and executing time-oriented queries on a massive, multi-parameter clinical information database poses significant computational challenges. The challenges encountered in converting high-resolution waveform and trend signals in the MIMIC II (Multi-parameter Intelligent Monitoring for Intensive Care II) database from an error-prone proprietary format to a stable open-source WFDB (Waveform Database) format is presented in the first half of this thesis. The design and implementation of a search engine that is capable of executing time-series queries on clinical information in the MIMIC II database such as lab results, medications, and nurse-verified values from bedside monitors is presented in the second half of this thesis. The search engine employs simple algorithms with little storage overhead to identify time periods in patient records that satisfy time series criteria based on thresholds and gradients of unevenly-sampled measurements. Results from queries executed on the search engine to detect physiological events of clinical interest were presented. Case studies on patient records returned as hits for queries were performed to review the strengths and limitations of the search engine.

Thesis Supervisor: Professor Roger G. Mark
Title: Distinguished Professor in Health Sciences and Technology
Professor of Electrical Engineering

Acknowledgments

I have received help from many people over the past one and a half year to complete this project. Without all their help and support, this project would not have been possible.

First of all, I would like to thank Professor Roger Mark for his wonderful guidance and support for my research at the Laboratory of Computational Physiology. I am extremely fortunate to have been guided by his insight in research and engineering throughout the course of this project.

I would like to thank Mohammed Saeed for all the help and thoughtful insight he contributed to this project. Mohammed's knowledge about the MIMIC II data was vital in the successful conversion of the data from Philips format to WFDB format. I also thank Mohammed for extracting the clinical information needed for the search engine from the relational database and putting them into MATLAB compatible format.

I would also like to thank Dr. Gari Clifford for all his help and support throughout this project. I received help from Gari at virtually every stage of this project. His keen insight helped me develop new perspectives in my research.

Special thanks to George and Benjamin Moody for for their help with the WFDB library. I am in awe at George's skill and experience in software engineering and biomedical signal processing. A big thank you to Benjamin for teaching me *gdb* and helping me find the worst software bug in my life thus far.

I would like to thank Dr. Thomas Heldt for his constant *'Thesis'* reminder and also his help in editing this thesis.

I also thank Dr. Brian Janz for his help and contribution to the search engine.

I thank Andrew Hung for developing an excellent web interface for the MIMIC II search engine. It was a lot of fun working with him and I wish him the best of luck in his college application and everything else.

I thank James Sun, Laurence Zapanta, Zaid Samar, Omar Abdala, Jennifer Shu, Margaret Douglass, Carlos Renjifo, and Tushar Parlikar for their friendship and support. Working at LCP was a fun and enjoyable experience also because of their companionship. Special thanks to James for our partnership through 6.021, 6.022 and the MATLAB expertise he shared with me. Also, I thank Carlos for *'the other version'* of my thesis.

Last but not least, I would like to thank my family, especially my parents, U Aung Tun and Daw Khin Myint Win, for their kind love, guidance and support. They have always been my source of inspiration and all my achievements today would not have been possible without their love and guidance.

This research work was supported by Grant R01 EB001659 from the National Institute of Biomedical Imaging and Bioengineering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

Today, medicare for patients with critical health conditions is provided in a in an specialized unit in a hospital known as the Intensive Care Unit (ICU). Patients who are admitted to an ICU are among the most critically ill in a hospital with life-threatening medical conditions that require constant monitoring and timely interventions in the event of deterioration. With the patient's life at stake, it is of utmost importance for the clinicians in the ICU to have simple, efficient ways of accessing and processing a patient's physiological data to make informed decisions regarding the patient's state and provide appropriate treatments.

1.1.1 Challenges and Issues in the ICU

Advances in computer and information technology coupled with the growth of sensor technology and computer networks have greatly increased the variety and complexity of medical devices available in a modern ICU. An ICU today is typically equipped with a number of bedside monitors that continuously record a series of waveform data such as the electrocardiogram (ECG), and blood pressure waveforms. Besides recording the waveform data, the monitors also record trends of the waveform data and generate alarms when the waveform data or the trend data fall out of acceptable ranges. In addition to the data generated by the bed-side monitors, clinicians also have access to clinical information systems with data from mechanical ventilators, laboratory tests, imaging studies and the notes compiled by other clinicians on the same patient.

Clinicians working in an ICU today are confronted with an overload in clinical information resulting from poor data organization and a poor interface to access data generated from a myriad of devices. It is customary to record time series data such as ECG signals at 256 Hz in today's ICU bedside monitors. Given that each monitor can record numerous time series signals (up to 4) and an ICU clinician on a 8-hour shift may need to process over 50 megabytes of data from the bedside monitor for a single patient. Since an ICU clinician may be responsible for more than 1 patient at the same time, the sheer volume of data generated by the monitors alone can be overwhelming for a person to keep track of and analyze in detail. The equipment in an ICU is typically composed of devices manufactured by companies with varying design goals. Hence, computer systems in an ICU built on top of a heterogeneous mixture of devices often result in systems with poor data integrity, organization and interface [1].

Information overload, coupled with poor data organization and integration, leads to deficiencies in patient care in an ICU. The continuous time series data from the bedside monitors are often poorly integrated with the related clinical information from other parts of the hospital systems, such as the laboratory reports and text notes [2]. Hence, it is often left to the clinicians to compile and process mentally the continuous data generated by the monitors and other intermittent data generated by the clinical information system in order to get a complete understanding of a patient's physical conditions. In an ICU, clinicians not only have to process accurately a tremendous amount of data under extremely tight time constraints, but also have to deal with the inherent noises and artifacts arising from poor data integration and organization. Dealing with all these challenges adds undesirable overhead to a clinician's work and can often lead to errors and delays of judgment resulting in the compromise of patient care. The study by Donchin et al. [3] noted that one major cause of human errors in the ICU is the difficulty in assessing patient state, which is directly related to the amount of overhead shouldered by clinicians in compiling and interpreting relevant patient information in the ICU.

In addition to generating a wealth of poorly organized and integrated information, the modern ICUs are also plagued by over-sensitive alarm systems. Most alarms in today's ICU are based on simple thresholds, meaning an alarm is triggered when the value of a physiological signal falls out of a predefined range of acceptable values. Such a simple alarm system is prone to generate excessive false alarms caused by non-physiological noise such as baseline wonder in a signal due to disconnected electrodes resulting from patient movements. The study by Lawless et al. [4] has shown that over 80% of alarms produced in a modern ICU are false positives. Since the omission of an alarm of a single life-threatening event is much more disastrous than generating a high number of false alarms, hospital alarm systems are set to be highly sensitive. However, an alarm system that produces a large number of false positives leads not only to wasted time and resources of clinicians but also to the neglect of truly dangerous events either through fatigue and desensitization to alarms or through the masking of real events by other false alarms [5].

1.1.2 Advanced Patient Monitoring System (AMS) and MIMIC II

The data issues and the over-sensitive alarm systems, coupled with a shortage of clinicians in the ICUs facing an increasingly aging and hence more fragile patient population [6] call for research and development of an Advanced Patient Monitoring System (AMS). An AMS facilitates a clinician's decision making process by generating intelligent alerts based on algorithms that analyze relevant patient information using advanced signal processing, modeling and classification techniques.

The Multi-parameter Intelligent Monitoring for Intensive Care II (MIMIC II) is the product of the initiative by the MIT Laboratory for Computational Physiology (LCP) to create a massive, temporal database to facilitate the research and development of an AMS[6]. The database contains comprehensive records of patient information available in the ICU, ranging from the continuous time series data and alarms generated by the bedside monitors and mechanical ventilators to laboratory reports, medication records, progress and discharge summaries compiled by clinicians, sufficient to replicate a patient's profile in the ICU. Presently, the MIMIC II database contains over 3500 patient records collected from multiple ICUs of a hospital in Massachusetts, occupying over 1 terabyte in disk space.

Due to the massive size and scope of the MIMIC II database, manually identifying regions or episodes of physiological interest among the huge number of patient records is an extremely laborious and time-consuming procedure. Therefore, a search engine that can automatically identify regions of physiological interest that meet a set of time series search criteria is necessary to improve the usability of the database. With slight modification, the algorithms in the search engine that identify notable regions in a time series physiological data stream can serve as a building block for future intelligent alert systems or any other AMS algorithm.

Another critical step in analyzing and distributing the MIMIC II data is to convert the data into a readily readable and searchable format. Over the last 20 years, the LCP has developed a robust, open-source data format, which supports annotations, known as the Wave Form Database (WFDB) format [9], and a wide array of signal processing tools to facilitate analysis of biomedical waveforms. The original time series data in MIMIC II were recorded in a proprietary data format developed by Philips Medical Systems. The Philips data schema was designed for simple archiving purposes and lacks proper support for research and development of advanced signal processing algorithms. Hence, significant effort was spent to convert the trend and waveform data in the MIMIC II database into an open source format.

1.2 Thesis Goals and Outline

This thesis is divided into two main parts. The first part of this thesis discusses the effort in converting the trend and waveform data in the MIMIC II database from the proprietary Philips format to an open source format. The second part of this thesis describes the development of a search engine that supports time series searches defined based on thresholds and gradients of clinical measurements in the records. There are 4 main objectives for this thesis:

- To describe the issues in converting the trend and waveform data in the MIMIC II database from a proprietary format to an open source format.
- To discuss critical issues in the current data collection process and suggest possible improvements.
- To explore MATLAB algorithms to perform time series searches on a sparse dataset efficiently and accurately .
- To discuss the limitations of the search engine and suggest future directions of research.

This thesis is organized in 4 chapters.

Chapter 2 describes in detail the proprietary data schema used to store the trend and waveform data in the MIMIC II database and the algorithms developed to convert the data from the proprietary format to an open-source format. The chapter also discusses unresolved problems in data conversion and the ongoing effort to create unified trend and waveform records for each patient. In addition, deficiencies in the current data collection process are reviewed and potential improvements are suggested.

Chapter 3 presents the design and implementation of a search engine for time series searches on an irregularly sampled sparse dataset. As currently implemented, the search engine supports time series searches with basic criteria defined based on thresholds and gradients of clinical data samples and complex criteria formed by combining the basic criteria with logical operators. The performance of the search engine is evaluated by performing searches to detect clinically interesting physiological events in patient records. A selected set of patient records detected by the search engine to contain physiological episodes of interest are presented as case studies to highlight the strengths and weaknesses of the search engine.

Conclusions to the data conversion project and the search engine project, together with suggestions for future work, are presented in Chapter 4.

Chapter 2

Data Conversion: from Proprietary to Open Source

2.1 Overview

The MIMIC II database includes two different types of data:

1. Data generated by bedside patient monitors including waveforms, trends and alarms.

2. Clinical information consisting of lab results, nurses' progress notes, intravenous (IV) medications, fluid balance, and patient demographics, and other relevant data from hospital archives.

As of August 2^{nd} 2005, the MIMIC II database contains time series data for over $3,500$ patients occupying over 1 terabyte in hard disk space. The LCP has also downloaded and incorporated into the database the clinical information of an additional $17,000$ patients.

Since the clinical information was downloaded as a standard Oracle database dump, there is no need to convert the database for clinical information into an open source format. However, time series data recorded from bedside monitors (trends and waveforms in particular) were stored in a proprietary and experimental Philips format that lacks a complete Application Program Interface (API) for easy and efficient access to data. Therefore, in order to achieve the goal of building an open-source database that can serve as the basis for research and development of an AMS, it is necessary to convert the time series data in MIMIC II to a format that allows fast and efficient access and searches of the data.

This chapter provides a brief overview of the data collection process for MIMIC II and a description of the source data schema for trend and waveform data. Then, Section 2.4 provides a brief introduction to the target data schema, the Waveform Database (WFDB) format. This description of is followed by a discussion of the problems discovered in data conversion and the algorithms developed to resolve those problems. The results of data conversion for MIMIC II and the continued challenges of merging time series data and mapping time series data with clinical information are then presented. Finally, some potential approaches to improve data collection for massive, multi-parameter databases are explored.

2.2 Data Collection

Figure 2-1 presents an overview of MIMIC II data collection process. Clinical information in CareVue [10], a proprietary clinical data repository consisting of lab tests, progress notes, and IV (intravenous) medications, is first archived in the Philips Information Support Mart (ISM), an Oracle-based relational database. The data in the ISM is then downloaded to a custom-built Oracle database via a Virtual Private Network (VPN) connection between the LCP and the hospital.

Data collection for time series data involves a number of intermediate steps, as shown in Figure 2-1. The process begins with storing data from the bedside monitors in a central database server known as the Philips Information Center Database Server (PICDBS). The data contain up to four continuously digitized signals such as ECG leads, ABP, PAP waveforms (Table 2.5) sampled at 125 Hz; up to 30 physiological trends such as HR, ABP_{sys}, ABP_{dias}, SpO_2 (Table 2.3) sampled at 1-minute intervals; alarms generated by the monitors and nurse central station; and indications of signal loss or failure to process data ("In-Ops"). The data from the PICDBS are then continuously retrieved and stored by a customized archiving agent in a proprietary data format developed by Philips. At approximately two-week intervals, the data from the archiving agent are manually downloaded to an external FireWire hard disk drive which is physically transported to the LCP. At the LCP, the data from the external hard disk are uploaded to RAID-equipped Linux file servers [6].

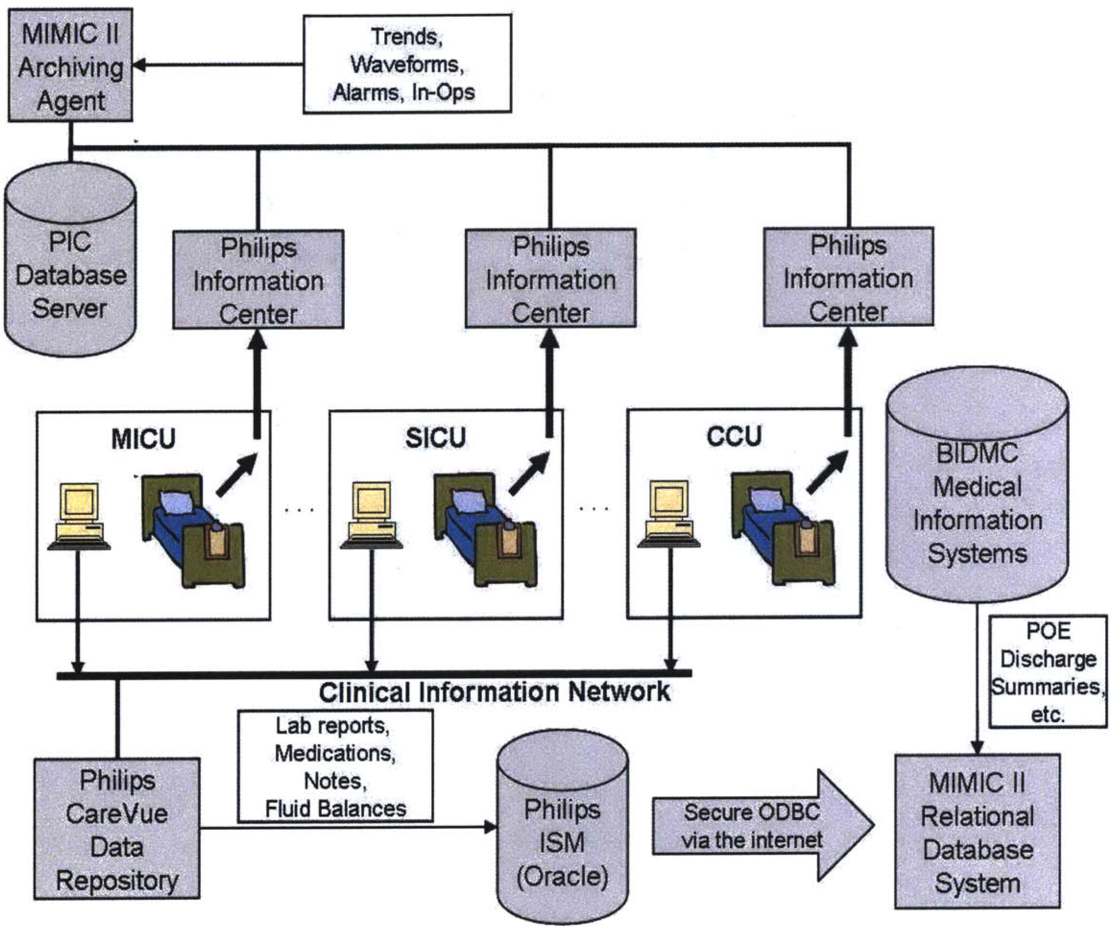

Figure 2-1: MIMIC II data collection process. Continuous physiological waveforms from bedside monitors are archived from the PICDBS by a customized archiving agent. Clinical information from the Philips CareVue repository is archived in the ISM and then downloaded via a VPN connection to a custom-built Oracle database [7].

2.3 Source Data Schema

The 2 main types of time series data in the MIMIC II database, the trend data and the waveform data, were stored in 2 different proprietary formats. While the trend data format is straightforward, the waveform format is significantly more complex as will be discussed below.

2.3.1 Trend Schema

The trend files in Philips format are named *ParamX.cfg*, where X is a four or five-digit case identification number for the patient. The schema for a trend file is shown in Table 2.1. Each trend file can be viewed as a table with 31 columns, the first column being the time stamp and each of the remaining columns being a trend, such as HR, ABP_{Sys}, or ABP_{Dias} (the complete list of which is shown in Table 2.3). Any missing data in the table is represented by a special value, -888.

Table 2.1: Data schema of a trend file. Each trend file is a table with 31 columns: the first column being the time stamp and each of the remaining 30 columns being a trend data (-888 = data not available).

Time Stamp	HR	ABP_{Sys}	ABP_{Dias}	...	...	CO
.	.	.	.	.	.	.
.	.	.	.	.	.	.
2001/08/08 05:33:55.345	70.00	125.540	85.830	.	.	-888.00
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Each time stamp in a trend file is an array of eight 16-bit unsigned short integers, each of which represents a time stamp attribute. All time stamp attributes are listed in Table 2.2. According to the schema, the time stamps are evenly sampled at $\frac{1}{60}$ Hz (once per minute) when available.

Table 2.3 lists all possible trend data types in their order of appearance in a Philips trend file. All trend data were stored as IEEE single-precision floating point numbers [11].

Table 2.2: Data Schema of a time stamp. Each attribute is a 16-bit unsigned short integer

Attribute	Description
1	Year
2	Month
3	Day of Week
4	Day
5	Hour
6	Minute
7	Second
8	Millisecond

2.3.2 Wave Schema

Waveform files in Philips format are named *WaveX.cfg*, where X is a four or five-digit case identification number for the patient. Figure 2-2 presents the schema for the first 3 minutes of a sample waveform file. In the example, there are 3 signals, each representing a physiological signal recorded by a bedside monitor at 125 Hz. A waveform file can contain up to 4 different signals. The signals are broken down into 1-minute segments and interleaved to form structures known as *WaveformRecords.* Hence, the 3-minute segment of signals shown in Figure 2-2 is stored as an array of 3 *WaveformRecords* in the wave file. A 1-minute segment of a signal within a *WaveformRecord* is called a *waveform snippet.* For each *waveform snippet* within a *WaveformRecord*, there is at least one *WaveInfo* structure that describes the *lead*, *gain*, and *baseline* values of the *waveform snippet.*

The *lead* of a signal is a term originated from the ECG and specifies the physical lead connected to a patient from which the signal is recorded. In the Philips waveform format, the concept of a *lead* is generalized to other waveforms such as ABP, so that the *lead* value is essentially a waveform identifier. Signal values are recorded in analog-to-digital converter (ADC) units, or *adus.* The *gain* for each signal specifies the number of *adus* corresponding to one physical unit, and the *baseline* value specifies the value of the ADC output that would map to 0 physical units at the input.

Clinical staff routinely change the *lead*, *gain*, and *baseline* values of bedside mon-

Table 2.3: All types of available trend data in a trend file. Each trend sample value is an IEEE single-precision floating point number.

Column	Trend	Description
1	N/A	Time Stamp
2	HR	Heart Rate
3	ABP_{Sys}	Arterial Blood Pressure (Systolic)
4	ABP_{Dias}	Arterial Blood Pressure (Diastolic)
5	ABP_{Mean}	Arterial Blood Pressure (Mean)
6	PAP_{Sys}	Pulmonary Artery Pressure (Systolic)
7	PAP_{Dias}	Pulmonary Artery Pressure (Diastolic)
8	PAP_{Mean}	Pulmonary Artery Pressure (Mean)
9	CVP	Central Venous Pressure
10	$PULSE$	Pulse Rate
11	$RESP$	Respiratory Rate
12	SpO_2	Saturation Périphérique en Oxygène (Pulse oximetry)
13	CO_2	Carbon Dioxide
14	ST_I	Standard ECG Lead I
15	ST_II	Standard ECG Lead II
16	ST_III	Standard ECG Lead III
17	ST_AVR	Augmented Unipolar Right Arm ECG Lead
18	ST_AVL	Augmented Unipolar Left Arm ECG Lead
19	ST_AVF	Augmented Unipolar Left Leg ECG Lead
20	ST_V1	Precordial (Chest) Lead V_1
21	ST_V2	Precordial (Chest) Lead V_2
22	ST_V3	Precordial (Chest) Lead V_3
23	ST_V4	Precordial (Chest) Lead V_4
24	ST_V5	Precordial (Chest) Lead V_5
25	ST_V6	Precordial (Chest) Lead V_6
26	NBP_{Sys}	Non-invasive Blood Pressure (Systolic)
27	NBP_{Dias}	Non-invasive Blood Pressure (Diastolic)
28	NBP_{Mean}	Non-invasive Blood Pressure (Mean)
29	$PAWP$	Pulmonary Artery Wedge Pressure
30	$SpO_{2,Aperiodic}$	Aperiodic Pulse Oximetry
31	CO	Cardiac Output

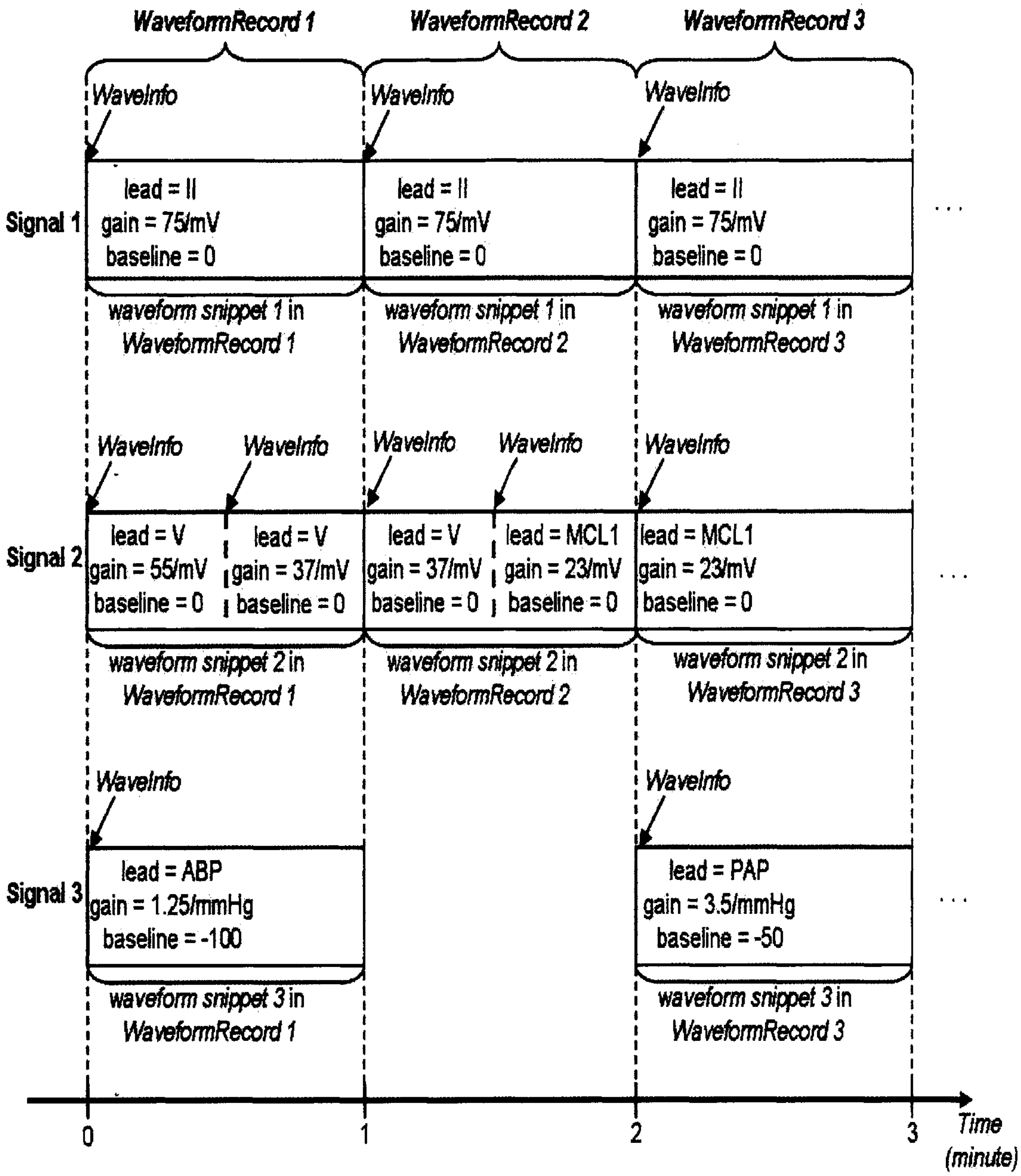

Figure 2-2: Snapshot of the first 3 minutes of a sample waveform file. The sample file contains three 125-Hz physiological signals. The signals are divided into 1-minute segments and then interleaved to form structures known as *WaveformRecords*. A 1-minute recording of a signal within a *WaveformRecord* is called a *waveform snippet*. Special structures, known as *WaveInfo* structures, specify the *lead*, *gain*, and *baseline* of a signal at any time within a *waveform snippet*.

itors to improve the visual display of physiological signals. Hence, the *lead*, *gain*, and *baseline* values of each signal can change dynamically at any time throughout a waveform file as shown in Figure 2-2. Each change in *lead*, *gain*, or *baseline* value is indicated by an additional *WaveInfo* structure within the corresponding *waveform snippet*.

The schema for a *WaveformRecord* structure is shown in Figure 2-3. Each *WaveformRecord* consists of a time stamp, specifying the starting time of the record, and up to 4 *waveform snippets*. The structure of a time stamp in a *WaveformRecord* is the same as that shown in Table 2.2. Each *waveform snippet* in turn contains 1 or more *WaveInfo* structures and a 1-minute snippet of data points. Each data point is an 16-bit integer value, holding an 8-bit resolution sample in the lower 8 bits. Each *WaveInfo* structure has a time stamp of its own and a *WaveSupportStruct*. A *WaveInfo* structure is added to a *waveform snippet* to update changes in the *lead*, *gain* and *offset* values of a signal at particular times within the *waveform snippet*. The structure of a time stamp in a *WaveInfo* structure is the same as that shown in Table 2.2.

The schema for a *WaveSupportStruct* structure is shown in Table 2.4. Some critical attributes of a *WaveSupportStruct* are *validFlag*, which indicates the validity of the *WaveSupportStruct*; *leadType*, which identifies the *lead* from which the sample values were recorded; and *markSizeMin, markSizeMax, markSizeValueMin,* and *markSizeValueMax*, which encode the *gain* and *baseline* values of the *signal* recorded by the monitor as follows:

$$gain = \frac{markSizeMax - markSizeMin}{markSizeValueMax - markSizeValueMin} .$$

$$baseline = \begin{cases} markSizeMin - 128 & \text{if } lead \text{ is } NOT \text{ ECG} \\ 0 & \text{otherwise} \end{cases}$$

Table 2.5 shows the known mappings from *leadType* to the actual lead types in an ICU bedside monitor.

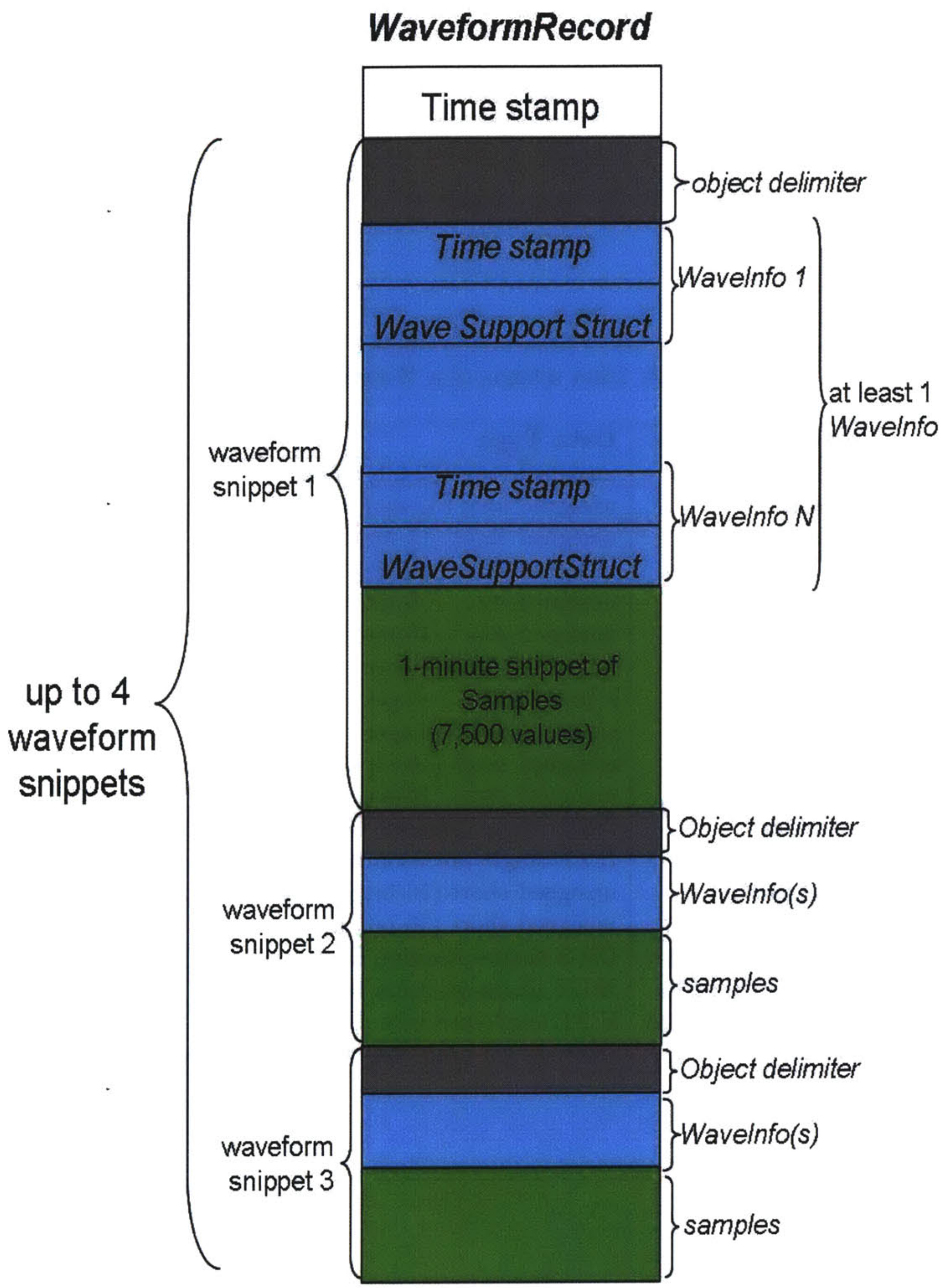

Figure 2-3: Data schema of a *WaveformRecord* structure. Each *WaveformRecord* consists of a time stamp and up to 4 *waveform snippets*. Each *waveform snippet* in turn contains 1 or more *WaveInfo* structures and a 1-minute snippet of sample values. Each *WaveInfo* consists of a time stamp of its own and a supplementary *WaveSupportStruct*.

Table 2.4: Data schema of a *WaveSupportStruct*.

Attribute Name	**Data Type**
validFlag	unsigned long (32-bit) integer
timeSec	unsigned long (32-bit) integer
timeMsec	unsigned short (16-bit) integer
leadType	unsigned short (16-bit) integer
bw	unsigned short (16-bit) integer
scaleLabel	unsigned short (16-bit) integer
scaleStatus	signed (32-bit) integer
calibrated	signed (32-bit) integer
validity	signed (32-bit) integer
sampleMin	unsigned short (16-bit) integer
sampleMax	unsigned short (16-bit) integer
valueMax	IEEE single-precision float
valueMin	IEEE single-precision float
markSizeMin	unsigned short (16-bit) integer
markSizeMax	unsigned short (16-bit) integer
markSizeValueMax	IEEE single-precision float
markSizeValueMin	IEEE single-precision float
samplingIntervalms	IEEE single-precision float
bwFreq	IEEE single-precision float array of length = 2

Table 2.5: Known mappings from *leadType* to actual lead types in an ICU monitor

leadType Value	Lead Type	Description
62101	*I*	Standard ECG Lead I
62102	*II*	Standard ECG Lead II
62103	*III*	Standard ECG Lead III
62104	*AVR*	Augmented Unipolar Right Arm ECG Lead
62105	*AVL*	Augmented Unipolar Left Arm ECG Lead
62106	*AVF*	Augmented Unipolar Left Leg ECG Lead
62107	*V*	Precordial (Chest) Lead V
62108	*V1*	Precordial (Chest) Lead V_1
62109	*V2*	Precordial (Chest) Lead V_2
62110	*V3*	Precordial (Chest) Lead V_3
62111	*V4*	Precordial (Chest) Lead V_4
62112	*V5*	Precordial (Chest) Lead V_5
62113	*V6*	Precordial (Chest) Lead V_6
62114	*MCL1*	Modified Chest Lead 1
62115	*MCL2*	Modified Chest Lead 2
62116	*MCL3*	Modified Chest Lead 3
62117	*MCL4*	Modified Chest Lead 4
62118	*MCL5*	Modified Chest Lead 5
62119	*MCL6*	Modified Chest Lead 6
62028	*RESP*	Respiratory Rate
62029	*PLETH*	Oximeter Plethysmographic waveform
62004	*ABP*	Arterial Blood Pressure
62010	*PAP*	Pulmonary Artery Pressure
62008	*CVP*	Central Venous Pressure

2.4 Target Data Schema

The Waveform Database (WFDB) format was chosen as the target data schema for the trend and waveform data in MIMIC II database for the following 3 main reasons:

- The WFDB library has been used by researchers worldwide for research and development of a wide variety of biomedical signal processing, display, analysis, and annotating applications over the past twenty years [12]. Hence, the existing set of open-source analysis libraries (particularly for the ECG) will facilitate the rapid analysis of the MIMIC data.

- WFDB, unlike the Philips proprietary format, is open-source and community-supported with backward compatibility in mind, and hence much easier to maintain and support for future applications.

- The LCP has extensive knowledge about the structure and design of the WFDB library.

2.4.1 WFDB Overview

The WFDB library, written in the C programming language, is a set of functions that provides easy and efficient storage and access to digitized, annotated signals stored in a variety of formats [13]. Although the WFDB library was originally designed for use with ECG databases, the library has now been expanded to support databases that include signals such as blood pressure, respiration, oxygen saturation, and the electroencephalogram (EEG). Today, the WFDB library is used by researchers worldwide for research and development of applications in biomedical signal processing. Over 60 freely available biomedical signal processing applications have been written with WFDB library support [12].

A WFDB database consists of a set of *records*, each of which contains a number of continuous time series signals for a single subject. Each WFDB *record* contains at least a *header* file and a *data* file. A *header* file contains information about the *record* including the beginning time stamp, the number of signals, the sampling frequency and the number of samples in the *record*. Additional information regarding individual signals in the record such as the *resolution*, *gain* and *baseline* values of each signal, and the name and format of the *data* file that contains the sample values of each signal is also specified in a header file. It is possible to store different signals belonging to the same record in multiple (possibly non-contiguous) *data* files in different formats. A *data* file stores sample values of one or more signals belonging to a single record in the format specified in the header file of the record. The signals in a record are assumed to be sampled at the same frequency throughout the duration of the record. Although the *gain* and *baseline* values for individual signals may be different, each signal is assumed to retain a constant *gain* and *baseline* values throughout individual *data* files in each record.

2.5 Data Conversion

There was no program available to read either trend or wave data in the proprietary formats. Only snippets of the header files that defined the data structures stored in trend and wave files were made available by Philips. Hence, the biggest challenge in converting MIMIC II trend and wave data was the handling of the discrepancy between the data schema specification provided by the vendor and the actual data recorded on disk.

Two separate programs, named *wfdbtrend* and *wfdbwave*, were developed to convert the trend and waveform data respectively. Each program accepts as input a trend or waveform file in Philips format and converts the input file to the WFDB format. The programs were developed, compiled and tested using GCC, the GNU C Compiler, on an IBM compatible Personal Computer running Fedora Core 3, a variant of the Linux operating system. The programs are located at [14] and are available for download with permission from the LCP. To compile the programs, one needs to first install and set up the WFDB libraries. For more information about installing and compiling applications with WFDB, please refer to the *WFDB Programmer's Guide* [9]. Two *bash shell* scripts, *convert-all-trend-data.sh* and *convert-all-wave-data.sh*, were written to automate the process of data conversion for trend and waveform data respectively.

2.5.1 Trend Conversion

Problems in Trend Schema

Although all time stamps in a trend file were expected to be 1 minute apart from one another by specification, it was discovered that over 80% of the records contain time stamps that were not exactly 1 minute apart. Furthermore, there were other time stamp problems such as invalid time stamps and missing or repeated time stamps.

Since all trend files consisted of the same number and type of signals and the unavailable samples were filled with a special value, −888, it was impossible to know beforehand which signals in a file contained actual data. During data conversion, about 2% of the trend files were found to contain no real data.

Since sample values were stored as IEEE single-precision floating point numbers in the proprietary trend files whereas WFDB requires sample values be written as integers, simply rounding off the floating point values into integer values could potentially cause loss of precision in sample values.

Algorithms for Trend Conversion

For each proprietary trend file, *ParmX.cfg*, a corresponding WFDB record, named *m2t_X*, is generated, where X is a 4 or 5 digit case identification number for a patient. Each WFDB trend record consists of a header file, named *m2t_X.hea*, and a data file, named *m2t_X.dat* file, in which the trend values were written as Format 16 integers (16-bit two's complement amplitude representation).

Trend conversion involves 3 steps as shown in Figure 2-4:

1. Read the entire trend file into memory, determining the earliest time stamp, latest time stamp and the signals that contain real data in the trend file.

2. Determine the sequence in which the sample rows are to be written into a WFDB file.

3. Create a new WFDB file and write only the signals that contain real data into the new file.

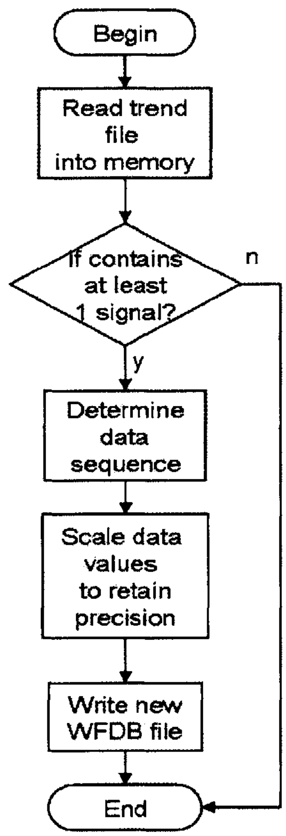

Figure 2-4: Data flow chart for trend conversion.

The algorithm to determine the sample sequence is shown in Figure 2-5. The algorithm involves the following 5 steps:

1. The entire trend file is read into memory and all time stamps are rounded to the closest minutes.

2. A *WFDB_Sequence* array of length: $(LatestTimeStamp - EarliestTimeStamp + 1)$ is allocated and all its elements are initialized with -1.

3. For each row i of samples in the trend file, determine its position in the *WFDB_Sequence* array, P_i, by finding $T_i - S$, where T_i is the time stamp of the row i and S is the earliest time stamp, and $T_i - S$ yields the number of minutes elapsed from S to T_i.

4. Write the row number R in the P^{th} element of the *WFDB_Sequence* array. If 2 rows have the same time stamp, the data in the row that comes later in the trend file will be written into the target WFDB file. For example, as shown in Figure 2-5, since both rows 4 and 6 have the same time stamp, only data from row 6 is written into the target WFDB file.

5. For each element in the *WFDB_Sequence* array containing a positive row number, R, write the R^{th} row of sample into the WFDB file. For each element in the *WFDB_Array* with the value -1, write a sample row filled with $WFDB\_null$ value, -32768, into the WFDB file. The *null* values from the original trend file, -888, are also converted into $WFDB\_null$ values for consistency.

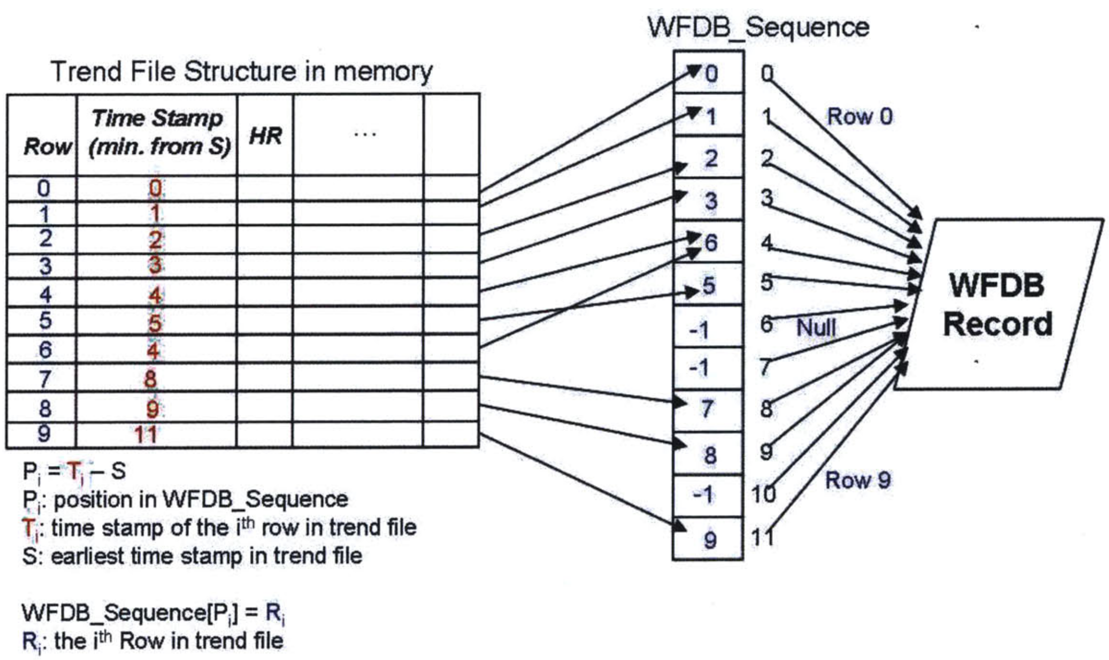

Figure 2-5: Graphical illustration of the algorithm to determine sequence in which rows of a trend file are written into a target WFDB file.

Sample values, stored as IEEE single-precision floating point numbers in a trend file, were converted to Format 16 in WFDB. To prevent loss of precision in sample values while converting from floating point numbers to integers, a gain value was assigned to each signal depending on its range of possible values. Sample values for each signal were amplified by their respective gain values before writing into WFDB to retain the required precision. The gain values of the (unscaled) units of all signals in a trend file are listed in Table 2.6.

2.5.2 Wave Conversion

Problems in Wave Schema

The Philips waveform files were also plagued by time stamp problems similar to those observed in trend files. However, there were fewer than 1% of waveform files that did not violate the time stamp schema specified by Philips. Due to the size of waveform files (over 1 gigabyte of disk space for some cases), it was not practical to read the entire waveform into memory and apply the sample sequence determination algorithm used for trend conversion. Hence, it was necessary to convert the waveform file minute-by-minute into the WFDB format.

The use of *WaveInfo* structures to identify each signal or indicate changes in the *lead*, *gain* and *baseline* values of a signal also introduced numerous unexpected problems. Although each signal was expected to contain at least 1 *WaveInfo* structure for identification purpose, there were often signals that contained no *WaveInfo*. Moreover, there were often cases where a signal contained duplicate or corrupted *WaveInfo* structures.

Although each one-minute segment of signal was expected to contain a fixed number ($125\frac{samples}{sec} \cdot 60sec = 7500samples$) of samples, on average, about 1% of segments in a wave file contained fewer or more than 7500 samples.

In the proprietary waveform file format, the *lead*, *gain* or *baseline* values of a signal can change dynamically at any time. On the other hand, a WFDB record requires the *lead*, *gain* and the *baseline* of a signal to remain fixed throughout the record.

Table 2.6: Gains and Units of Physiological Signals in a Trend File

Column	Trend	Gain	Unit (unscaled)
1	*Time Stamp*	N/A	N/A
2	HR	10	beats per min
3	ABP_{Sys}	10	mmHg
4	ABP_{Dias}	10	mmHg
5	ABP_{Mean}	10	mmHg
6	PAP_{Sys}	10	mmHg
7	PAP_{Dias}	10	mmHg
8	PAP_{Mean}	10	mmHg
9	CVP	10	mmHg
10	$PULSE$	10	beats per min
11	$RESP$	10	per min
12	SpO_2	100	%
13	CO_2	100	%
14	ST_I	100	mV
15	ST_II	100	mV
16	ST_III	100	mV
17	ST_AVR	100	mV
18	ST_AVL	100	mV
19	ST_AVF	100	mV
20	ST_V1	100	mV
21	ST_V2	100	mV
22	ST_V3	100	mV
23	ST_V4	100	mV
24	ST_V5	100	mV
25	ST_V6	100	mV
26	NBP_{Sys}	10	mmHg
27	NBP_{Dias}	10	mmHg
28	NBP_{Mean}	10	mmHg
29	$PAWP$	10	mmHg
30	$SpO_2\_Aperiodic$	10	%
31	CO	100	Lit per min

The most challenging problem observed in a Philips waveform file was a byte-skipping error, in which an odd number of bytes were inserted or skipped at a certain point in a file, making all subsequent content of the file unreadable. Over 20% of Philips waveform files contained one or more occurrences of such a byte-skipping error.

Algorithms for Wave Conversion

For each Philips wave file, *WaveX.cfg*, a set of corresponding WFDB records, named *m2w_X_N*, is generated, where X is the 4 or 5 digit case identification number for a patient and N can potentially range from 0 to the total number of *WaveInfo* structures in a Philips wave file. Each WFDB wave record is made up of a header file, named *m2w_X_N.hea*, and a data file, named *m2w_X_N.dat* file.

Figure 2-6 presents the top-level flow chart for the waveform conversion program from Philips format to WFDB format. In each iteration of the algorithm, a 1-minute segment of the waveform file, known as the *WaveformRecord* by Philips terminology, is read into memory. Any *waveform snippet* in a *WaveformRecord* without at least one valid *WaveInfo* structure is discarded. Each *WaveInfo* structure within a *waveform snippet* is rigorously validated and any corrupted or duplicate *WaveInfo* structures are discarded. If the number of samples for a minute-long waveform snippet is less than 7500, the missing samples are filled in by WFDB *null* values at the end of the segment. If the number of samples exceeds 7500, only the first 7500 samples are retained. Since sample values contain only 8-bit resolution data, each sample value is converted from a signed 16-bit integer value in a wave file to Format 80 (an 8-bit offset binary representation) in WFDB.

Consecutive *WaveformRecords* are written to a single WFDB record until 2 consecutive *WaveformRecords* contain time stamps that are not exactly 1-minute apart or a change in the *lead, gain*, and *baseline* of a signal occurs within a single *waveform snippet*. Therefore, a single *waveform snippet* can potentially be converted into numerous WFDB records and converting a single proprietary wave file can potentially result in a large number of WFDB records.

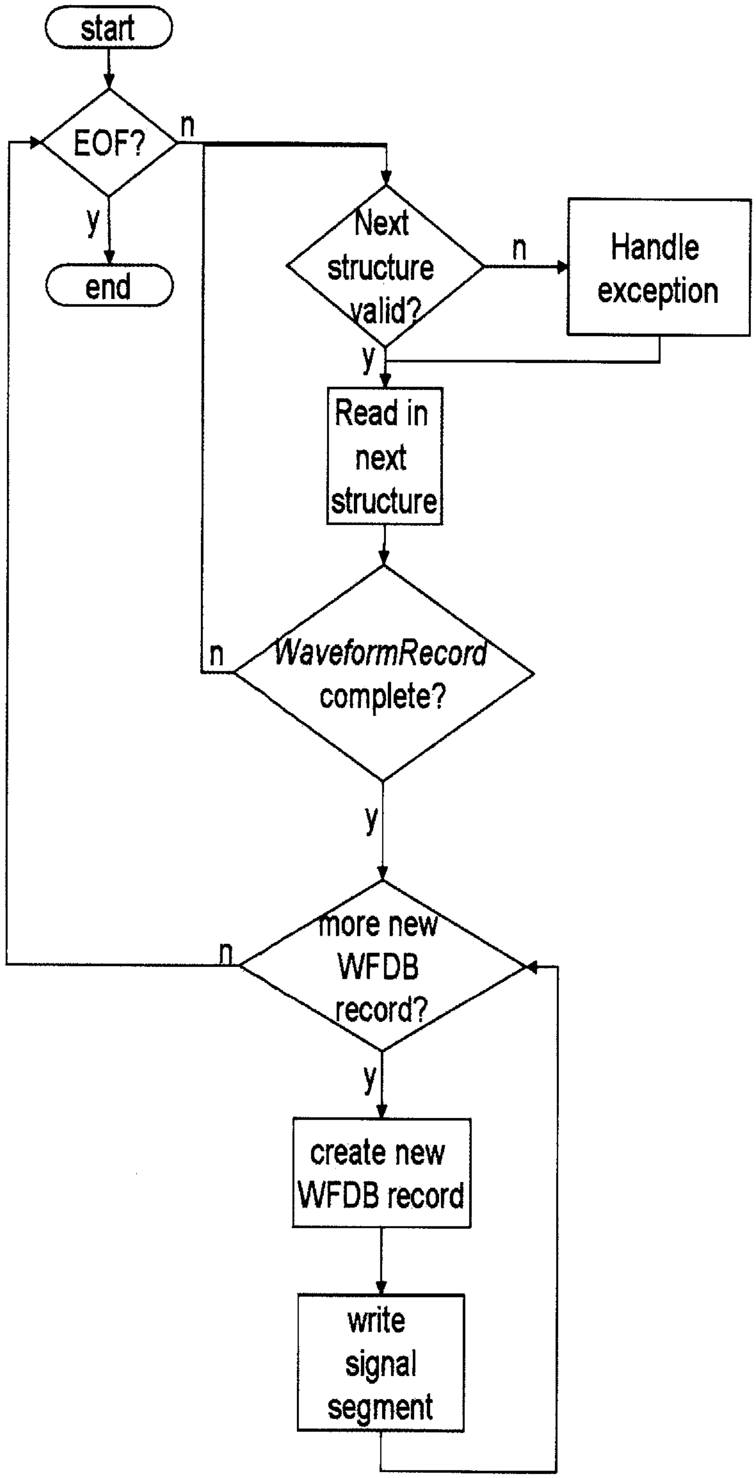

Figure 2-6: Data flow chart for wave conversion.

The regular conversion algorithm is interrupted whenever a byte-skipping error is encountered. By trial and error, it was found that the file pointer needs to be rewound by at least 51 bytes in order to successfully recover a byte-skipping error. Therefore, in the event of a byte-skipping error, the file pointer is rewound to 51 bytes before the point of failure and the data is scanned byte-by-byte until a known data structure such as a time stamp or an object delimiter is identified. Once a known data structure is found, the regular conversion algorithm is resumed.

2.6 Data Conversion Results and Data Verification

2.6.1 Data Conversion Results

Table 2.7 presents the results from converting MIMIC II trend and wave files from their original proprietary formats to WFDB format. The data conversion programs successfully converted all trend files with at least 1 valid sample value and all wave files with at least 1 valid *WaveformRecord*. It is worth noting that about 6% of trend files (220 files) contained no valid sample and about 7% of wave files (122 files) contained not a single *WaveformRecord*.

Table 2.7: Summary of Data Conversion Results

Data Type	No. of 'non-empty' Source Files	No. of Successful Conversions	Success Rate (%)
Trend	3190	3190	100%
Wave	1579	1579	100%

2.6.2 Data Verification

To verify the correctness of the data conversion algorithm, 15 random cases were chosen and the data from the trend files for those cases were cross-verified with those from the wave files. The cross-verification process for each case involves the following 5 steps:

1. Select 10 random time points in a WFDB trend file.
2. For each time point selected in step 1, find the corresponding WFDB wave file containing the time point.
3. Open up a visual image of the WFDB wave file segment of interest in *Wave*, a WFDB signal visualizing tool [12].
4. Estimate the heart rate from the RR intervals of the ECG leads available from the plot generated by *Wave* and cross-verify with the heart rate recorded in the trend file.
5. If ABP and/or PAP signals are available for the wave file segment, cross-verify the visual estimates of systolic and diastolic ABP and/or PAP values observed in the plot generated by *Wave* with the corresponding ABP_{sys}, ABP_{dias}, PAP_{sys} and PAP_{dias} values recorded in the trend file.

The results from data verification showed that the values in trend files for all time points selected for verification were matched by the corresponding values in wave files with less than 2% error.

2.7 Continued Challenges

2.7.1 Segmentation of a Wave File into multiple WFDB records

A single wave file in the proprietary format can be segmented into multiple WFDB records during conversion (Figure 2-7). The segmentation may be due to irregularly-spaced time stamps, or changes in the *lead*, *gain* or *baseline* values of one or more signals in the wave file. Such segmentation could add undesirable overheads to applications designed to process a continuous stream of WFDB data for a single subject. The latest release of the WFDB library developed by George and Benjamin Moody deals with the problem of segmentation by creating a *meta-header* file that combines multiple WFDB wave segments belonging to the same case and allows applications to access a single *meta-record* associated with the *meta-header* file for a single case.

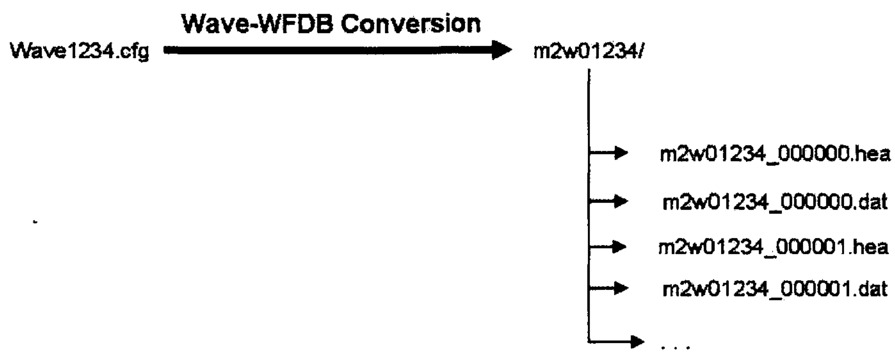

Figure 2-7: Segmentation of a single proprietary wave file into multiple WFDB records during Wave-WFDB conversion.

2.7.2 Identical Cases in Different Date Directories

The source files for time series data, including trend and wave files were organized in directories named according to their *data transfer dates* in *YY-MM-DD* format (Figure 2-8). On a *data transfer date*, all time series data archived in the PICDBS (Figure 2-1) were downloaded to an external hard disk which was physically transported to the LCP where the files were uploaded to RAID-equipped Linux file servers. All downloaded time series data files were then purged from the PICDBS.

For a patient who was still hospitalized in an ICU on a *data transfer date*, his or her time series data recorded up to the *data transfer date* were downloaded in a wave file named *Wave_X.cfg* and a trend file named *Param_X.cfg*, where X is the case identification number for the patient. After the trend and wave files were downloaded and purged from the PICDBS on the *data transfer date*, the recording of time series data for the patient is resumed in newly created *Wave_X.cfg* and *Param_X.cfg* files. Those newly created *Wave_X.cfg* and *Param_X.cfg* files would be transferred to the file server on a subsequent *data transfer date*. Hence, wave and trend files with identical case identification numbers can exist under multiple *data transfer date* directories.

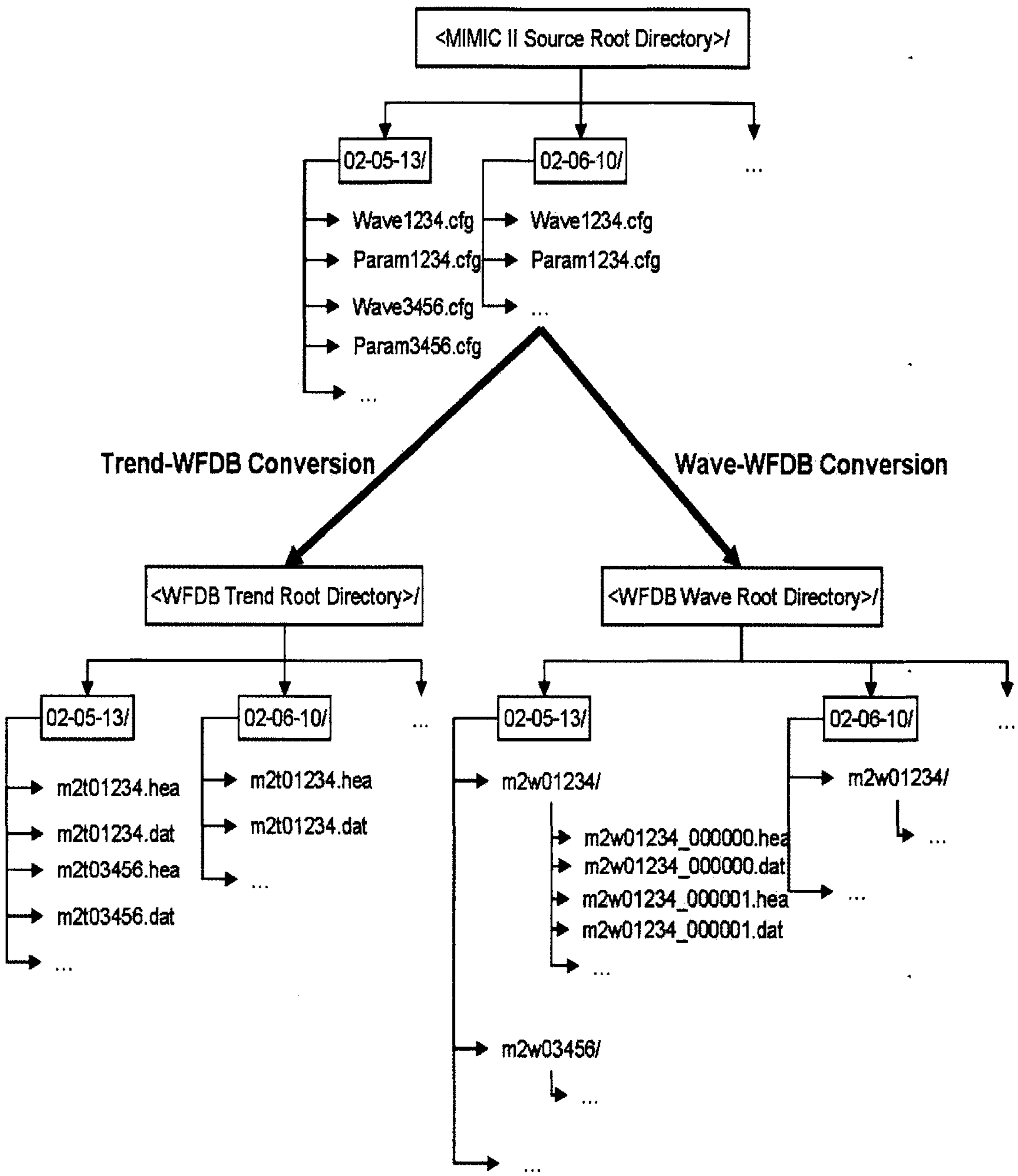

Figure 2-8: MIMIC II time series data organization. The files are stored in directories named after the *data transfer dates*. Case 1234 appears in two directories because the data for the case was partially downloaded on two different *data transfer dates*.

This phenomenon is clearly illustrated by the patient with case identification number 1234 under the directories: $02-05-13/$ and $02-05-27/$ in Figure 2-8. Such *virtual segmentation* caused by constraints in data transfer adds undesirable overheads to applications expecting a continuous stream of data from WFDB. The *meta-header* file feature in the next release of WFDB is designed to handle this *virtual segmentation* problem transparently from WFDB applications in a manner analogous to the way it handles the wave file segmentation problem discussed in the previous section.

2.7.3 Mappings from Cases to Patients

The MIMIC II time series data recorded by the bedside monitors in an ICU are identified by a case identification number. On the other hand, the clinical information from the CareVue repository is identified by a patient identification number. A patient identification number assigned by the hospital to a patient is unique for each patient. A case identification number is assigned to a bed by the monitoring system when a patient is admitted to an ICU. Hence, a patient could be assigned multiple case identification numbers if he or she is admitted multiple times to a single ICU or different ICUs during his or her stay in the hospital. Furthermore, if the case identification number is not reset properly when a new patient is admitted, the recording for a single case identification number could contain physiological data recorded from multiple patients. For over 40% of the patient identification numbers in the current MIMIC II database, the corresponding case identification numbers were not properly matched because the clinical staff in the ICU failed to input for each case the medical record number which serves as the link between the case and the patient record in the clinical information repository. LCP has initiated an effort to detect reliably the mappings between the patient identification numbers and the case identification numbers using the availability of different physiological signals for different patients and the correlation between nurses' verified data stored in the CareVue repository and the trend data recorded by the bedside monitors.

2.7.4 Unknown Leads in Wave Files

Table 2.5 lists all the known mappings from the monitor manufacturer's *lead type identification numbers* to physiological signal leads. During data conversion, it was discovered that the proprietary wave files contain recordings from some unknown *lead type identification numbers.* Such signals were stored as *unknown* signals in WFDB. Those *unknown* signals can potentially be identified with the help of expert clinicians or advanced signal identification algorithms.

2.8 Suggested Improvements

Most problems related to time series data in the MIMIC II database were caused by the limitations in physical equipment, the constraints in the data collection process and the design shortcomings in the proprietary data format used to store the data. Based on the experience with MIMIC II data conversion, the following 4 steps are suggested to improve future data collection projects.

1. Human intervention should be minimized in data collection. The time spent by a researcher traveling back and forth to download data represents wasted man-hours in research effort. Real-time, automated electronic data transfer via a VPN is a much more efficient and economical solution that also guarantees a more timely and systematic data transfer.

2. A single identification number should be used consistently for each subject in data collection. If multiple identification numbers need to be used, great effort should be taken into establishing and maintaining mappings between different identification numbers. Poor mappings between multiple identification numbers can lead to potential loss in data as well as wasted time, and energy in retrospectively remapping the data.

3. Only completed cases should be retrieved and removed from the repository to avoid *virtual segmentation* of data records caused by data transfer.

4. In any data collection effort, it is essential to use a data format that preserves data integrity and allows systematic and efficient access to data. Using a poorly designed data format can potentially result in great losses in data, time, money and other valuable research resources.

Chapter 3

Search Engine for a Massive, Multi-parameter Clinical Information Database

3.1 Overview and Motivation

Given the massive size and scope of the MIMIC II database (over 3000 patient records, consisting of over 200 signals totalling over 1 terabyte of data), it is impractical to manually examine records in the database and identify those containing interesting physiological events. Therefore, it is necessary to have a search engine for the database that allows efficient search and retrieval of patient records containing episodes of physiological interest. An ideal search engine for a massive multi-parameter biomedical database should be able to respond to not only simple queries on patient demographics and information recorded in text notes but also sophisticated time-oriented multi-dimensional queries on waveform and trend recordings from bedside monitors as well as other clinical information including laboratory results, medications and fluid balances.

Searches on signals from bedside patient monitors such as ECG and trend recordings are difficult not only because of the massive amount of data involved but also because of the inherent susceptibility of those signals to noise and the difficulty in describing and measuring the 'similarity' between signal features. Therefore, as a first step towards building a comprehensive search engine for the MIMIC II database, a simplified search engine was built to handle time-oriented, multi-dimensional queries on the clinical information consisting of computerized laboratory results, medications and nurses' verified values recorded from the bedside monitors. By limiting searches to such variables, noises and artifacts are greatly reduced.

Excluding from the search engine the high-resolution, noise-prone waveform and trend data from bedside monitors does not necessarily preclude the detection of interesting physiological events from patient records. Patient records with episodes of physiological interest can be identified by searching through only computerized lab results, medications and human-verified clinical data. For example, a potential way to detect episodes of acute renal failure is to look for patient records whose creatinine level exceeds 2 and whose rise in creatinine trend level exceeds 200% within 2 days. In this example, there are 2 different types of queries involved:

1. A threshold query that looks for all creatinine measurements greater than 2.

2. A gradient query that looks for changes in creatinine level that exceed 200% within 2 days.

The MIMIC II search engine described in this chapter is designed to handle both threshold and gradient queries as well as more complex queries formed by joining these 2 types of queries using logical operators 'AND', 'OR' and 'NOT'.

Section 3.2 provides an overview of the existing technologies for indexing and searching time series data, and discusses the need to explore new techniques to efficiently conduct time-oriented queries. Section 3.3 then describes the primary objectives of the MIMIC II search engine design. The design of the search engine's underlying data structures is presented in Section 3.4. The algorithms developed to process the data structures are discussed in Section 3.5. The web-based interface developed for the search engine using the MATLAB Webserver is presented in Section

3.6. Finally, the chapter concludes by describing the performance of the search engine and its strengths and limitations in Section 3.7.

3.2 Existing Technologies for Indexing and Searching Time Series Data

Time-oriented or temporal queries cannot be efficiently implemented in traditional relational databases and SQL. Therefore, over the past 20 years, there has been active research on the design and implementation of temporal databases [15]. However, as presented in the survey by Chomicki et al. [16], temporal query languages such as TQuel [17], and TSQL2 [18], not only have limited ability to express time intervals using cumbersome syntax and semantics but also lack support for multi-parameter time series data with different temporal resolutions.

As an alternative to temporal query languages, there has also been extensive research on indexing time series databases using dimensionality reduction techniques and handling time-based queries by searching on those indices. Some well-explored dimensionality reduction techniques include the Discrete Fourier Transform (DFT) [19, 20], Singular Value Decomposition (SVD) [21], the Discrete Wavelet Transform (DWT) [20, 22, 23], and various approximation techniques [24, 25]. However, the DFT assumes stationarity of the signal whereas SVD requires linearity in the combination of the components of the signals. SVD also imposes strict orthogonality between the components which may not necessarily be independent. The DWT requires linear combination of wavelets while imposing significant overhead for storing and indexing wavelet indices. The choice of wavelet in the DWT is often arbitrary as is the relevant scale or approximation level. The approximation techniques, on the other hand, require sophisticated data structures and algorithms while also imposing tremendous overhead in storage requirement. Saeed et al. [26] proposed storing a selected set of precomputed wavelet coefficients in a relational database to handle time-based queries efficiently. Although this method outperforms conventional SQL approach by 2 orders of magnitude for trend queries on precomputed wavelet coefficients, it does not provide a satisfactory solution to queries that require wavelet coefficients that have not been precomputed. Above all, there is no conclusive proof that any dimensionality reduction techniques can outperform an efficiently implemented program that conducts a simple, exhaustive search on gradients in terms of accuracy and efficiency. The design and implementation of a search engine that handles temporal queries by the use of an algorithm that efficiently computes and searches an exhaustive set of gradients for a given set of samples is presented in the remaining sections of this chapter.

3.3 Design Objectives

The MIMIC II Search Engine was designed with 4 primary objectives.

1. The program is designed to serve as a filtering tool for researchers interested in investigating patient records that meet specific pathophysiological criteria. Due to the tremendous amount of data and the wide range and variety in the types of patient records available in the MIMIC II database, the search engine is designed to accommodate the need of researchers who need to focus on a specific type of patient. Careful consideration was taken into designing a search engine that provides researchers with maximum flexibility in specifying criteria.

2. The simple, yet efficient, algorithms used in the search engine to handle temporal queries were developed to serve as a basis of comparison for more sophisticated techniques for temporal searches.

3. The program was designed to be a building block for a more comprehensive search engine. Therefore, modularity of components was an important factor in design considerations for the search engine to ensure compatibility with future extensions.

4. The data structures, functions and algorithms developed for the MIMIC II search engine were designed to be as generalizable as possible so that they can

be extended for use in future signal processing, data mining or machine learning applications.

The search engine was implemented in MATLAB 7 [27], a high-level computer programming langauge with extensive libraries for signal processing, signal visualization and numerical methods. MATLAB was chosen for its comprehensive set of built-in libraries and efficiency for software development and debugging. The choice of MATLAB also ensures portability because software written in MATLAB can be compiled into an operating system-specific executable. Furthermore, since the WFDB_tools [28] package provides an interface between MATLAB and the WFDB library, building the search engine in MATLAB ensures the ability to extend the implementation to bedside monitor data in the future.

3.4 Data Structure Design

The MIMIC II search engine, as implemented, is limited to searching lab results, medications, demographic information and nurse-verified data downloaded from the bedside monitors. All those data were originally downloaded and stored in a standard relational database format as described in 2.2. Due to the overhead involved in writing MATLAB programs to directly query a relational database, a selected set of 2048 patients that contain clinical data of interest was downloaded and organized into MATLAB compatible structures which will be described in detail in the following sections.

3.4.1 Clinical Information Structures

127 clinical data items for 2048 patients together with their demographic information (age, sex and patient identification number (pid)) were retrieved and stored in data structures (a-d) described below. Since MATLAB 7 can efficiently handle its data structures in single precision floating point numbers much more efficiently than double precision floating point numbers, without biasing any calculations, all data values were

converted into single precision floating point numbers whenever the loss in precision is negligible (i.e. $< 1\%$). All unknown values in the data structures are represented by NaN (Not a Number) since MATLAB can efficiently operate on vectors containing indeterminant elements (NaN).

a) Demographics

The *demographics* matrix stores the demographics information (pid, age and gender) of the patients in the database. Table 3.1 shows the schema of the *demographics* matrix. Ages of patients that exceed 89 years are all rounded to 89.9 to ensure deidentification of protected health information [29]. In the sex column, male is represented by 1 and female is represented by 2. The rows of the matrix are sorted by pid in ascending order.

Table 3.1: Schema of the *demographics* matrix. *Demographics* has 3 columns (pid, age and sex). Age above 89 years are rounded to 89.9. Male is represented by 1 and female is represented by 2. The rows are sorted by pid in ascending order. NaN indicates unknown.

pid	age	sex
.	.	.
5	55	2
22	64	1
23	89.9	NaN
.	.	.

b) columnMappings

Table A.1 in Appendix A shows the content of *columnMappings*, a cell array that describes the type of items stored in each column of *pidCareVue* matrix (discussed in the next section).

c) pidCareVue

The *pidCarevue* matrix stores values of nurse-verified data from the bedside monitors, and ventilators, lab results, and medications. Table 3.2 shows the schema of the *pidCarevue* matrix. Each column of the matrix holds a different type of value specified by the corresponding cell in *columnMappings* (Table A.1 in Appendix A). For example, column 1 in pidCarevue holds the *pid* while column 3 holds the *CVP* of patients.

Table 3.2: Schema of the *pidCarevue* matrix. Each column of the matrix holds a type of value specified by the corresponding cell in *columnMappings* (Table A.1). $T =$ (the number of 5-minute intervals elapsed from the earliest time stamp + 1). The values of T for each patient are not regularly sampled. Rows in *pidCarevue* are sorted by pid in ascending order. Rows with the same pid are sorted by T in ascending order.

pid	**T**	**cvp**	...	**dopaminedrip**
.	.	.	.	.
123	27	32.0	.	NaN
123	33	28.0	.	NaN
127	1	44.0	.	2.00
.	.	.	.	.

Each row in *pidCarevue* stores all available data for a particular patient at a particular time point as single precision floating point numbers. Rows in *pidCarevue* are sorted by pid in ascending order. Rows with the same pid are sorted by T in ascending order. Since not all types of data are available for a patient at a particular time point, many of the positions in *pidCarevue* are filled with NaN, which indicate the data not being available. Despite the redundancy compared to a traditional relational database schema, the data schema for *pidCarevue* allows very fast and efficient retrieval of a specific type of data for a particular patient. For example, to retrieve heartrate for patient with pid 123, one only needs to access the third column in *pidCarevue* for the rows with pid 123.

The time stamp T_i for row i in *pidCarevue* is given by $T_i = \lfloor \frac{t_i}{12} \rfloor + 1$, where t_i is the time in hours since the beginning of the patient record and hence T_i is rounded to the nearest 5 minute time period.

The 5-minute interval was chosen because the highest sampling frequency for the data in the CareVue repository was $\frac{1}{300}Hz$. It should be noted that the time stamps T for each patient in the *pidCarevue* matrix are not separated by regular intervals.

d) pidTimeLine

Table 3.3 shows the schema of the *pidTimeLine* matrix. The second column of the matrix stores the starting time stamp of the patient with the pid in first column in a serial date number format returned by the MATLAB *datenum* function. Since converting the serial date number into single precision results in a significant precision loss, all values in the matrix were stored as double precision floating point numbers. The third column of the matrix stores the latest T observed in *pidCarevue* (Table 3.2) for each patient. Hence, the third column of *pidTimeLine* is essentially the observed duration of each patient record. For each searchable pid, there is a row of entry in the matrix. The rows in the matrix are sorted by pid in ascending order.

Table 3.3: Schema of the *pidTimeLine* matrix. The first colum of the matrix holds the list of pids in *pidCarevue*. The second column stores the earliest time stamp of each patient observed in the database in a MATLAB serial date number format. The third column stores the latest T observed in *pidCarevue* for each patient. The rows are sorted by pid in ascending order.

pid	start time	length of record/duration
.	.	.
123	731065.791666667	973
127	731066.312500000	330
133	731066.833333333	61
.	.	.

3.4.2 Supplementary Data Structures

The following three supplementary data structures, (*pidIndices, pidSampleCount,* and *pidGradientStat*), were designed to improve the performance of the search engine by providing information to improve data access or screen out records that do not meet search criteria.

a) pidIndices

The *pidIndices* matrix holds information that allows fast and efficient access to a patient's data in the *pidCarevue* matrix. Table 3.4 describes the schema of *pidIndices*. The matrix contains 3 columns, the first of which holds the list of pids in *pidCarevue*. The second and third columns hold the beginning and ending row numbers that contain data for the corresponding pid in *pidCarevue*. For example, since the beginning row number is 515 and the ending row number is 540 for pid 123, one only needs to retrieve data from rows 515 to 540 to access any data for patient with pid 123. For each searchable pid, there is a row of entry in the matrix. The rows in *pidIndices* are sorted by pid in ascending order.

Table 3.4: Schema of the *pidIndices* matrix. The first colum of the matrix holds the pids. The second and third columns hold the beginning and ending row numbers in *pidCarevue* for the corresponding pid. The rows are sorted by pid in ascending order.

pid	beginning row	ending row
.	.	.
123	515	540
127	541	590
133	591	604
.	.	.

b) pidSampleCount

The *pidSampleCount* matrix stores information regarding the number of real samples (all samples excluding the NaNs) for each item that a patient record contains. The search engine uses sample count to screen out records that cannot meet a specified criteria. For instance, if the search is to look for patients with heart rates greater than 110, a patient without any heart rate sample should never be considered as a potential hit for the criteria. If the search is to look for patients whose heart rate drops by more than 20% within 1 hour, patients with fewer than 2 heart rate samples should never be considered as potential hits because a change in heart rate is only meaningful if there are more than 1 sample of heart rate data.

Table 3.5: Schema of the *pidSampleCount* matrix. The first column of the matrix holds the list of pids and each column from the second column onwards holds the list of the sample counts of an item for the corresponding patients in column 1. The columns were arranged according to the mappings specified in *columnMappings*. The rows in *pidIndices* are sorted by pid in ascending order.

pid	**T**	**cvp**	...	**dopaminedrip**
.	.	.	.	.
123	26	20	.	0
127	50	33	.	1
133	6	5	.	0
.	.	.	.	.

The data schema for *pidSampleCount* is shown in Table 3.5. The first column of *pidSampleCount* holds the list of pids in *pidCarevue*. Each column from the second column onwards holds the list of the sample counts of an item for the corresponding patients in column 1. For example, one knows that there are 20 samples of CVP for patient record with pid 123 since the third column in the row with pid 123 holds a number 20 in Table 3.5. The columns were arranged according to the mappings specified in *columnMappings*. For each searchable pid, there is a row of entry in the matrix. The rows in *pidIndices* are sorted by pid in ascending order.

c) pidGradientStat

pidGradientStat is a 3-dimensional matrix that stores precomputed values to facilitate searches for temporal queries. *pidGradientStat* can be thought of as a stack of six 2-dimensional matrices with the same dimensions and schema as *pidSampleCount* but storing different information. The values are stored to help screen out patient records that will not satisfy a given trend query. The types of values stored in the 6 matrices are listed in Table 3.6, in the order in which they were stacked.

Figure 3-1 shows an example of how the 6 value types stored in *pidGradientStat* can be derived from a series of sample values of an item in a patient record. In the example, the longest interval length (dT_{max}) is 8 while the shortest interval length (dT_{min}) is 1. The most positive change in CVP values occured from $T = 1$ to $T = 5$

Table 3.6: Description of values stored in each of the six 2-dimensional matrices in *pidGradientStat*.

Stack Number	Value Type	Description
1	dX_{min}	the most negative change (in regular units) in measurement level of an item in a patient record
2	dX_{max}	the most positive change (in regular units) in measurement level of an item in a patient record
3	dT_{min}	the minimum interval length (in units of 5-minute intervals) over which a change in value can be calculated for an item in a patient record
4	dT_{max}	the maximum interval length (in units of 5-minute intervals) over which a change in value can be calculated for an item in a patient record
5	dP_{min}	the most negative change (in percentage) in measurement level of an item in a patient record
6	dP_{max}	the most positive change (in percentage) in measurement level of an item in a patient record

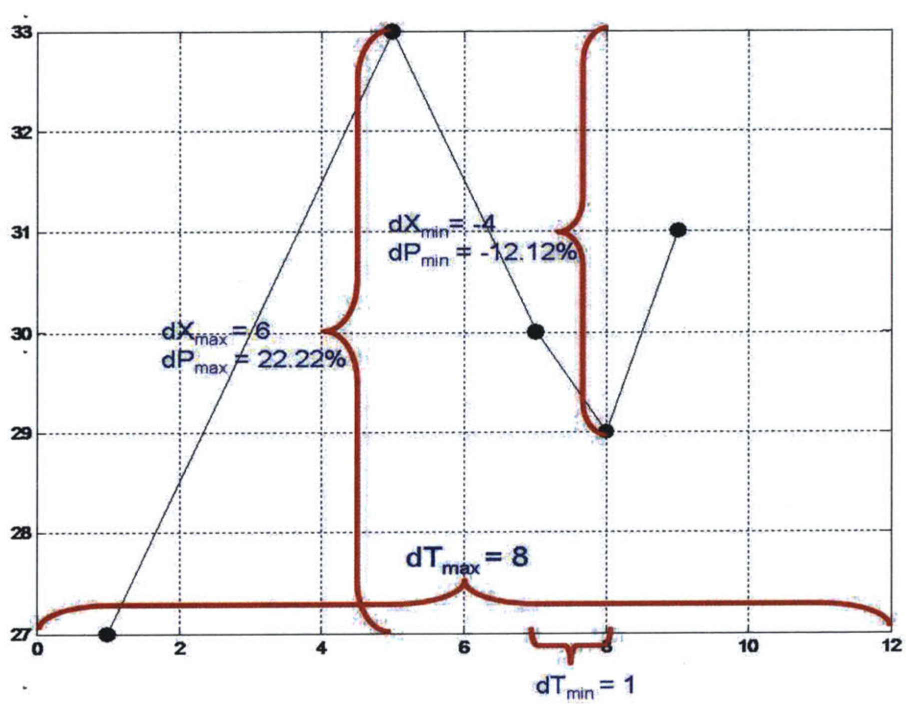

Figure 3-1: Example showing how 6 value types stored in *pidGradientStat* are derived.

representing $dX_{max} = 33 - 27 = 6$ and $dP_{max} = \frac{33-27}{27} = 22.22\%$. The most negative change in CVP values occured from $T = 5$ to $T = 8$ representing $dX_{min} = 29 - 33 = -4$ and $dP_{max} = \frac{29-33}{33} = -12.12\%$.

3.4.3 Input/Output Data Structures

The MIMIC II search engine requires as input 2 strings of characters, one specifying the demographic criteria and the other specifying the time series search criteria. The search engine returns the search results as an array of *search hit structures* as will be explained in this section.

a) Demographic Query String

An example of a character string specifying a set of demographic criteria is given below:

[[age 50 70 & sex 1 1] | ~[pid 100 nan]]

The above query searches for male patients with ages between 50 AND '&' 70 years old (inclusive) OR '|' (patients with pid NOT '~' greater than or equal to 100.

Each demographic query string is required to satisfy the following constraints:

- A demographic query string is made up of a set of basic demographic criteria joined together by binary logical operators & (*and*) and | (*or*) or the unary negation operator ~.

- A basic demographic criterion (eg. *age 50 70*) consists of 3 components: a property (age, sex or pid), a lower bound and an upper bound. For sex, male is represented as 1 and female as 2.

- For each basic demographic criterion, the search engine will return patient records whose property values lie within the lower bound and the upper bound. For example, given the criterion *age 50 70*, the search engine will return patients whose $age \geq 50$ and $age \leq 70$.

- For the gender criterion, the upper bound and the lower bound are required to be identical. For example, it is invalid to specify *sex 1 2* as a demographic criterion. The overhead of an extra integer in specifying the gender criterion 'sex 1 1' is to allow the query parsing algorithm to handle the gender criterion as a normal demographic criterion.

- Either the upper bound or the lower bound in a demographic criterion can be specified as NaN. If a bound is specified as NaN, the constraint for the bound is relaxed. For example, given the criterion *age 50* NaN, the search engine will return patients whose $age \geq 50$.

- Each binary logical operator requires 2 arguments, each of which could be a basic demographic criterion or a set of demographic criteria joined together by logical operators. A pair of square brackets is required to specify the scope for each binary logical operator.

- Any subset of a demographic query string can be negated using $\sim$, the negation operator. A pair of square brackets is required to specify the scope over which the $\sim$ operator is to be applied.

b) Time Series Query String

For a time series query, the search engine identifies not only the patient records that satisfy the criteria but also the time points at which the criteria are satisfied.

The structure of a time series query string is very similar to that of a demographic query string. An example of a character string specifying a set of time series criteria is given below:

(heartrate 120 NaN NaN NaN 0 & artbpmean NaN -20 NaN 6 1)

The above query searches for periods in patient records where heart rate equalled or exceeded 120 beats per minute and mean arterial blood pressure dropped by 20% or more within intervals of less than 6 hours.

Each time series query string is required to satisfy the following constraints:

- A time series query string is made up of a set of basic time series criteria joined together by binary logical operators & (*and*) and | (*or*) or the unary negation operator $\sim$.

- Each basic time series criterion consists of 6 components. However, there are 3 different types of basic time series criteria which are distinguishable by the content of the components.

  1. *Threshold search criterion*: The schema of a threshold search criterion is as follows:

     $$item\ value_{min}\ value_{max}\ NaN\ NaN\ 0.$$

     Item is a searchable item the complete list of which is available in Appendix A. $Value_{min}$ and $value_{max}$ are the lower and upper bounds for the *item* of interest. The last 3 arguments in the query string, NaN, NaN and 0 in a threshold search criterion are placeholders that allows the query parsing algorithm parse a threshold search criterion in the same way as a gradient search criterion. Hence, given the criterion '*heartrate 120 NaN NaN NaN 0*', the search engine will identify patient records with samples of heart rate $\geq$ 120 and the time period during which the criterion was satisfied.

  2. *Gradient search criterion (by value)*: The schema of a gradient search criterion (by value) is as follows:

     $$item\ \Delta value_{min}\ \Delta value_{max}\ \Delta T_{min}\ \Delta T_{max}\ 0.$$

     Item is a searchable item as before. $\Delta value_{min}$ and $\Delta value_{max}$ are the lower and upper bounds for the change in values in item of interest whereas ΔT_{min} and ΔT_{max} are the smallest and largest time intervals (in hours) for which the change in values should be calculated. For example, given the criterion' *artbpmean NaN -20 NaN 6 0*', the search engine will identify time periods of less than 6 hours in which the mean arterial blood pressure dropped by more than 20 mmHg.

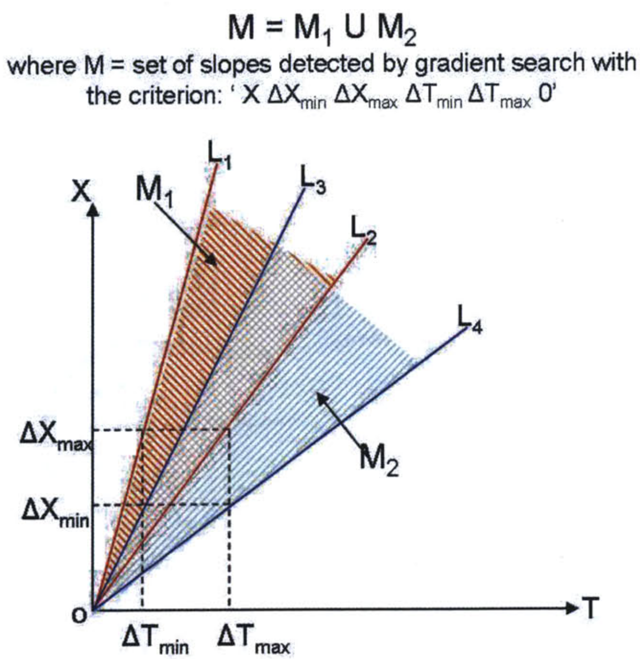

Figure 3-2: Bounds on the rate of change of X for a gradient search criterion: 'X ΔX_{min} ΔX_{max} ΔT_{min} ΔT_{max} 0'.

Figure 3-2 provides a graphical illustration of the set of slopes M that is detected by a gradient search with the criterion: 'X ΔX_{min} ΔX_{max} ΔT_{min} ΔT_{max} 0'. As shown in the figure,

$$M = M_1 \cup M_2$$

where M_1 = gradients bounded between $L_1 = \frac{\Delta X_{max}}{\Delta T_{min}}$ and $L_2 = \frac{\Delta X_{max}}{\Delta T_{max}}$ and M_2 = gradients bounded between $L_3 = \frac{\Delta X_{min}}{\Delta T_{min}}$ and $L_4 = \frac{\Delta X_{min}}{\Delta T_{max}}$

3. *Gradient search criterion (by percent)*: The schema of a gradient search criterion (by percent) is as follows:

$$item\ \Delta value_{min}\ \Delta value_{max}\ \Delta T_{min}\ \Delta T_{max}\ 1.$$

The only difference between this criterion and the previous one is that the $\Delta value_{min}$ and $\Delta value_{max}$ are interpreted as the lower and upper bounds of percentage changes in item values and the final argument (flag) is set to 1. For example, given the criterion '*artbpsys -50 -90 1 6 1*', the search engine will identify time periods of less than or equal to 6 hours, but greater than 1 hour during which the systolic arterial blood pressure dropped by at least 50% but not more than than 90%.

- Binary logical operators, & and |, and the unary negation operator $\sim$ can be used in a time series query string in the same way as a demographic query string. However, in a time series query string a pair of parentheses (instead of square brackets) is required to specify the scope of the operators.

c) Search Hit Structure

Table 3.7 shows the schema of a search hit structure. The structure consists of 4 attributes:

- *Pid* specifies the patient identification number.
- *Start* specifies the starting time of the patient record in MATLAB serial date format.
- *Onset* contains a set of time points (in units of 5-minute intervals from the starting time) that mark the beginning points of the periods during which the time series search criteria are satisfied.
- *Offset* contains a set of time points (in units of 5-minute intervals from the starting time) that mark the ending points of the periods during which the time series search criteria are satisfied.

If only the demographic criteria are specified, the *start*, *onset* and *offset* attributes in the search hit structure are all filled with the special value NaN.

3.5 Algorithm Design

Figure 3-3 presents the top-level flow chart for the MIMIC II search engine. As shown in the figure, the search engine processes each query in 3 major steps:

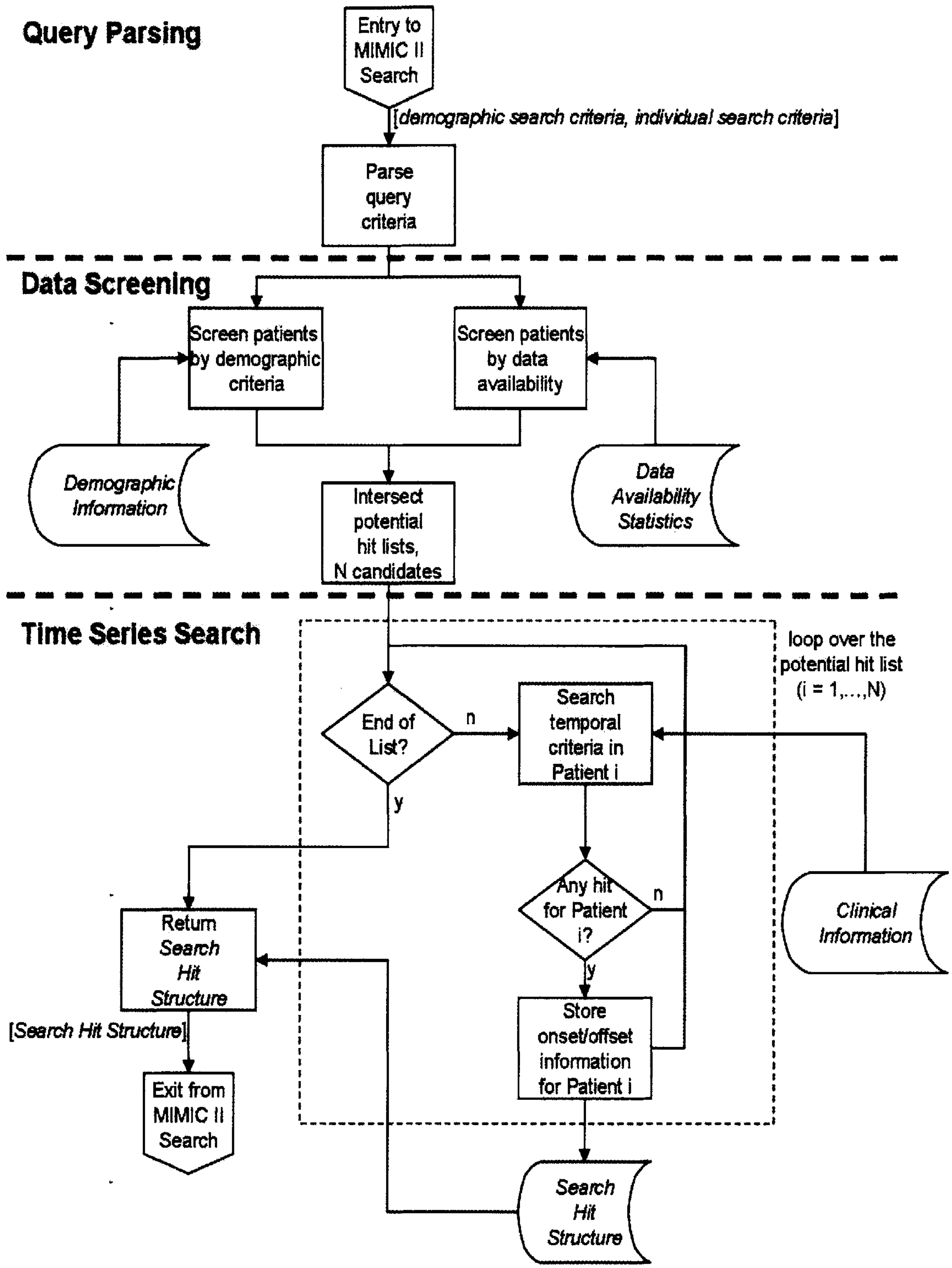

Figure 3-3: Flow chart for the MIMIC II search engine.

Table 3.7: Data schema of a search hit structure.

Attribute	Description	Example
pid	patient identification number	123
start	starting time of patient record (in MATLAB serial time unit)	731065.791666667
onset	a set of time points (in units of 5-minute intervals from start) marking the beginning of periods during which time series search criteria are satisfied	[1 33 55]
offset	a set of time points (in units of 5-minute intervals from start) marking the end of the periods during which time series search criteria are satisfied	[5 34 60]

1. Query Parsing,

2. Data Screening, and

3. Time Series Search.

The program takes as input 2 query strings: one specifying the demographic criteria and the other specifying the time series search criteria. The query strings are parsed into internal data structures that specify the steps required to fulfill the given query. The query parsing algorithm is described in Section 3.5.1.

After parsing the search criteria, the program proceeds to narrow its search space by screening out records that could not meet the criteria to form potential hit lists. The program generates a potential hit list based on the demographic criteria and another potential hit list based on the time series criteria. The data screening algorithms are described in Section 3.5.2. The 2 potential hit lists are then intersected to form a single potential hit list for the given query.

The search program then iterates through each patient record in the potential hit list to determine the time points at which the individual search criteria were met. For each hit, i.e. a patient record that contains at least a period of time during which the individual search criteria were met, the information regarding the hit is stored in a search hit structure. Finally, the search engine terminates by returning its results in an array of search hit structures.

3.5.1 Query Parsing Algorithm

Figure 3-4 provides a graphical illustration of how a demographic query string is converted by the query parsing algorithm into internal query specification structures. There are 4 query specification structures which together specify the steps to be taken by the search engine to satisfy the query and the parameters required for each step:

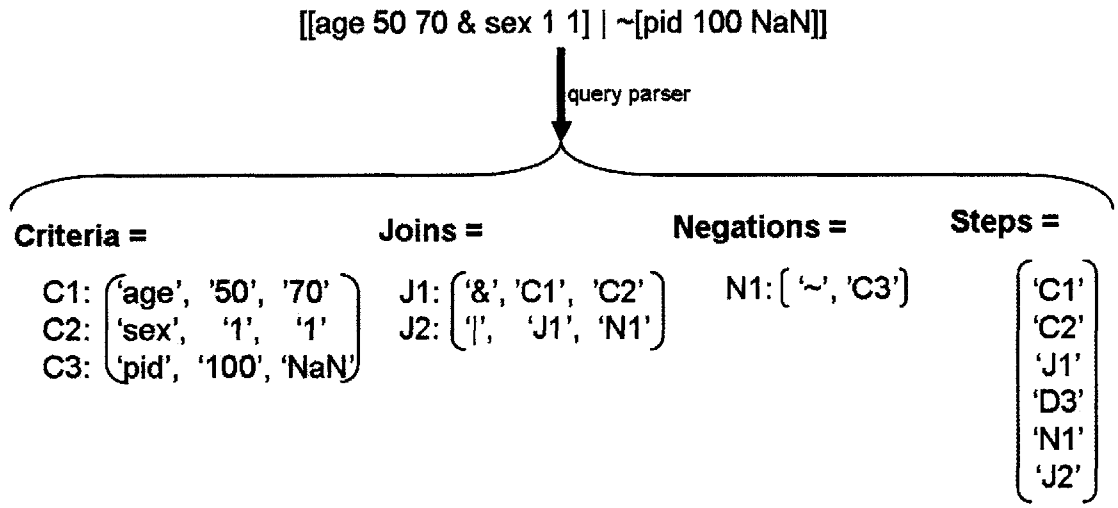

Figure 3-4: Graphical illustration of query parsing. The demographic query string: '[[age 50 70 & sex 1 1] | ~[pid 100 nan]]' is converted into 4 internal query specification structures: *Criteria, Joins, Negations* and *Steps.*

- The *Criteria* matrix holds the set of basic criteria in a query string. The structure of the *Criteria* matrix for a demographic query string is different from that for a time series query string due to the difference in the number of components required to specify a basic criterion in the two different query strings. The *Criteria* matrix of a demographic query string holds 3 components whereas that of a time series query string holds 6 components.

- The *Joins* matrix holds the information required to execute the binary logical operations (& and |) as specified in a query string.

- The *Negations* matrix holds the information required to execute the unary negation operation (~) as specified in a query string.

- The *Steps* array specifies the sequence of steps required to carry out the search specified in a query string.

The structures for the *Joins, Negations* and *Steps* matrices are the same for both demographic query strings and time series query strings.

The same query parsing algorithm is used to parse both demographic query strings and time series query strings. Figure 3-5 shows the flow chart of the query parsing algorithm used to parse both types of query strings. Query parsing is carried out in the following steps:

1. Initialize *Criteria, Joins, Negations* and *Steps* to empty vectors and matrices. Initialize 2 empty stacks: *Stack1*, *Stack2*. Initialize internal variables: $i = j = k = 0$.

2. Break a query string into components: operators '$\&, |, \sim$', criteria, and scope specifiers '( ) [ ]'.

3. For each component of the query string, take the following action:

   - If the component is an operator, push it to the top of *Stack1*.
   - If the component is a valid basic criterion, add it to *Criteria*; increment i by 1; push 'C_i' to the top of *Stack2* and add 'C_i' to *Steps*.
   - If the component is an end scope specifier ') or]', take the following action:
     (a) Pop (remove from top of *Stack*) the top operator (Op) from *Stack1*.
     (b) If Op is a binary operator '& or |', pop 2 items P_1, P_2 from the top of *Stack2*; add ['Op','P_1','P_2'] to *Joins*; increment j by 1; push 'J_j' to the top of *Stack2* and add 'J_j' to *Steps*.
     (c) If Op is a unary operator '$\sim$', pop 1 item P_1 from the top of *Stack2*; add ['Op','P_1'] to *Negations*; increment k by 1; push 'N_k' to the top of *Stack2* and add 'N_k' to *Steps*.

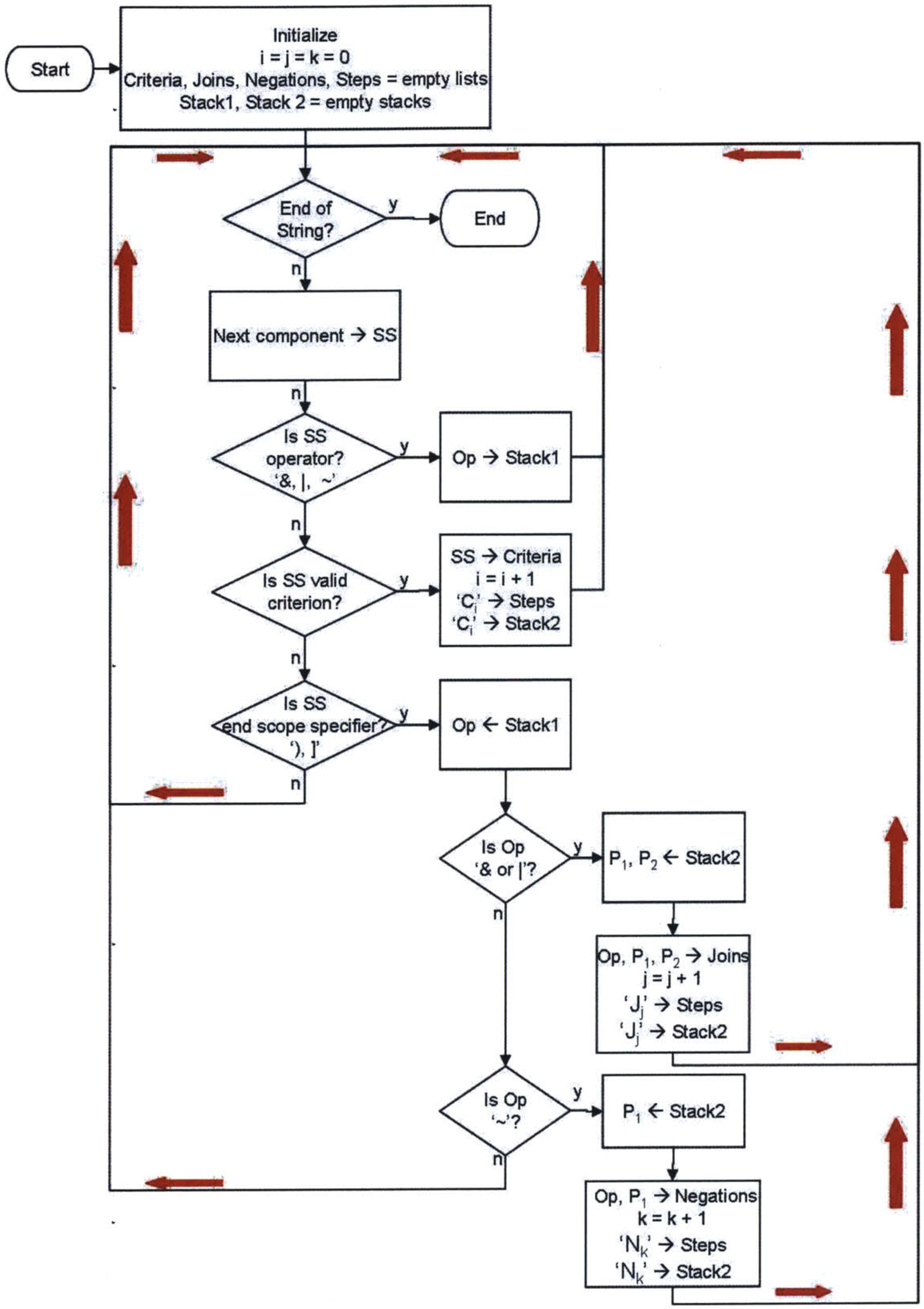

Figure 3-5: Flow chart of the query parsing algorithm.

3.5.2 Data Screening Algorithms

The primary purpose of data screening is to reduce the number of records for which the search engine needs to search for time series criteria because time series searches involve intensive computation that can slow down the performance of the search engine.

Patient records are screened by demographic criteria if a demographic query string is specified for a search. Data screening by demographic criteria is performed as follows:

- For each basic demographic criterion, a potential hit list that contains patient records that satisfy the criterion is generated.
- For each & operator in the query string, a set union operation is applied to the 2 potential hit lists generated for the 2 arguments of the operator.
- For each | operator in the query string, a set intersection operation is applied to the 2 potential hit lists generated for the 2 arguments of the operator.
- For each $\sim$ operator in the query string, the potential hit list generated for its argument is subtracted from the set of all pids to form a new potential hit list.

Patient records are also screened based on time series search criteria. However, data screening for time series search criteria is complicated by the $\sim$ operator. The & and | operators in a time series query string can be handled in the same way as in a demographic query string for data screening. However, the $\sim$ operator in a time series query string cannot be handled in the same way as in a demographic query string because the result of applying the $\sim$ operator to a potential hit list and that of not applying the $\sim$ operator to the list are not mutually exclusive. For example, assuming that patient record 123 is a potential hit for the query searching for patients with heart rate greater than 120 beats per minute, 123 cannot be removed as a potential hit for the query searching for patients with heart rate *not* greater than 120 beats per minute because patient 123 may in fact contain heart rate samples with fewer than

120 beats per minute. Therefore, different data screening strategies were implemented to handle query strings with and without a $\sim$ operator.

If a time series query string that does not contain any $\sim$ operator, a potential hit list for the query string is generated in the following steps:

- For each threshold search criterion, only patients that contain at least one sample that meets the criterion is included in the potential hit list for that criterion.
- To find the potential hit list for each gradient search criterion, first find the misses (patient records that do not meet the criterion) using the following 4 conditions:

$$dT_{max} < \Delta T_{min}$$

$$dT_{min} < \Delta T_{max}$$

$$dX_{max} < \Delta X_{min}(\text{in sample unit}) \text{ or } dP_{max} < \Delta X_{min}(\text{in percentage})$$

$$dX_{min} < \Delta X_{max}(\text{in sample unit}) \text{ or } dP_{min} < \Delta X_{max}(\text{in percentage})$$

  where $dT_{max}, dT_{min}, dX_{max}, dX_{min}, dP_{max}$, and dP_{min} are values stored in *pidGradientStat* for the item in the criterion and $\Delta T_{min}, \Delta T_{max}, \Delta X_{min}$, and ΔX_{max} are constraint values specified in the gradient search criterion. All patient records identified as misses are then eliminated from the set of all searchable patient records to get the potential hit list.
- Set union and set intersection operations are then performed on pairs of potential hit lists as specified by the & and | operators in the time series search string to get the final potential hit list.

For a time series query string that contains at least a $\sim$ operator, the following steps are performed to get the potential hit list:

- For each threshold search criterion, only patients with at least 1 sample of the item specified in the criterion are included in the potential hit list for the criterion.

- For each gradient search criterion, only patients with at least 2 samples of the item in the criterion are included in the potential hit list for the criterion.
- Set union and set intersection operations are then performed on pairs of potential hit lists as specified by the & and | operators in the time series search string to get the final potential hit list.

3.5.3 Time Series Search Algorithms

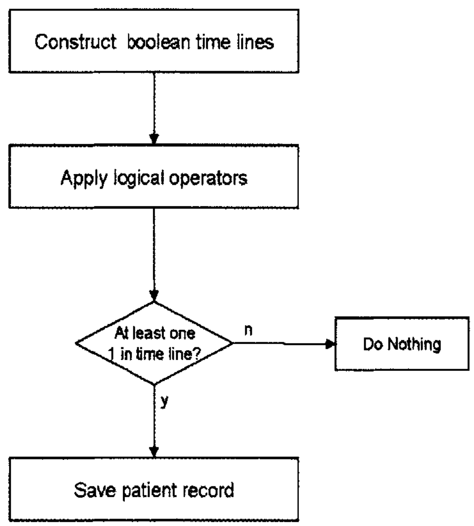

Figure 3-6: Steps of a time series search on a single patient record.

The goal of a time series search on a patient record is to identify periods of time in the patient record where the time series criteria specified in a query string are satisfied. Figure 3-6 shows the 3 steps carried out for a time series search on a single patient record:

1. For each basic time series criterion (either a threshold search or a gradient search), construct a time line of evenly sampled boolean values, where the 1's in the time line indicate the sample values of the item specified in the criterion at those time points satisfy the given criterion. It is worth noting that all boolean time lines constructed for a single patient are evenly sampled and identical in length to ensure logical operators (& and |) can be applied directly to join

together a pair of boolean time lines constructed for different searchable items belonging to the same patient. The length of a boolean time line for a patient record is the the observed length of the patient record stored in the third column of the *pidTimeLine* matrix.

2. Apply logical operators, &, |, and ~, to the boolean time lines constructed in the previous step. The operators are applied to the time lines in the order specified in the time series query string to get the final boolean time line that identifies time points where all the criteria specified in the query string are satisfied.

3. If the final boolean time line contains at least a single 1, the patient record is saved in a *search hit structure* with the time line for the patient being converted from the evenly sampled boolean string representation to the onset, offset points representation.

a) Threshold Search

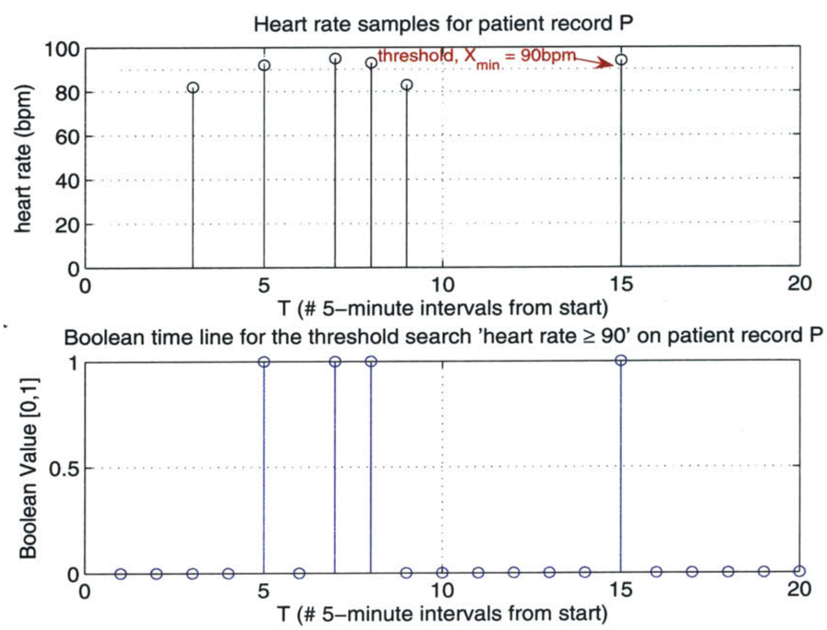

Figure 3-7: Example of a threshold search with $X_{min} = 90bpm$.

Figure 3-7 illustrates an example of a threshold search of 'heartrate $\geq$ 90 beats per minute' for patient record P. The threshold search is carried out as follows:

1. Assuming that the observed length of P is 20 according to the *pidTimeLine* matrix, a boolean time line of length 20 is constructed with all values initialized with 0's.

2. For each sample value in P that satisfies the criterion, the corresponding boolean value in the boolean time line is set to 1.

b) Gradient Search

A gradient search is carried out in the following 5 steps:

1. Compute $\Delta\mathbf{X}$, a matrix that contains all possible time-oriented changes in sample values and $\Delta\mathbf{T}$, a matrix that contains all possible time intervals between sample values.

2. Find a set of two dimensional index pairs in $\Delta\mathbf{X}$, $\mathbf{I_{ij}^{\Delta x}}$, where $\Delta X_{min} \leq \Delta x \leq \Delta X_{max}$ and a set of index pairs in $\Delta\mathbf{T}$, $\mathbf{I_{ij}^{\Delta t}}$ where $\Delta T_{min} \leq \Delta t \leq \Delta T_{max}$. Find the intersection of the two sets of index pairs to obtain $\mathbf{I}_{ij}^{hits}$, which contains the set of index pairs that mark the periods during which the gradient search criterion is satisfied.

3. If $\mathbf{I}_{ij}^{hits}$ contains index pairs in which multiple row indices are paired with the same column index, retain only the index pair with the smallest row index.

4. If $\mathbf{I}_{ij}^{hits}$ contains index pairs in which multiple column indices are paired with the same row index, retain only the index pair with the highest column index.

5. Construct a boolean time line based on the remaining indices.

The gradient search algorithm is explained below with a step by step gradient search with criterion: *'heartrate -20 0* $\frac{1}{12}$ $\frac{3}{4}$ *0'* on an example patient P. The criterion searches for episodes in P where the change in heart rate values (Δx) falls between -20 and 0 within time intervals (Δt) $\frac{1}{12}$hour and $\frac{3}{4}$hour. Hence, according to the criterion,

$\Delta X_{min} = -20, \Delta X_{max} = 0, \Delta T_{min} = \frac{1}{12} * 12 = 1$, and $\Delta T_{max} = \frac{3}{4} * 12 = 9$. It is worth noting that although ΔT_{min} and ΔT_{max} were specified in hours at input, the values were converted into units of 5-minute intervals used for internal computation.

The heart rate samples **X** and the corresponding time stamps **T** of P are provided in Table 3.8

Table 3.8: Heart rate samples **X** and the corresponding time stamps **T** of an example patient record P.

T (# 5-minute intervals from start)	X (heart rate samples in bpm)
3	82
5	92
7	95
8	93
9	83
15	94

Step 1: Given $\mathbf{X} = \begin{pmatrix} 82 & 92 & 95 & 93 & 83 \end{pmatrix}$, $\Delta\mathbf{X}$ can be computed as follows:

1. Form square matrices, **shiftMat** $= \begin{pmatrix} 82 & 92 & 95 & 93 & 83 & 94 \\ 82 & 92 & 95 & 93 & 83 & 94 \\ 82 & 92 & 95 & 93 & 83 & 94 \\ 82 & 92 & 95 & 93 & 83 & 94 \\ 82 & 92 & 95 & 93 & 83 & 94 \\ 82 & 92 & 95 & 93 & 83 & 94 \end{pmatrix}$ by replicating **X** and **baseMat** $= \begin{pmatrix} 82 & 82 & 82 & 82 & 82 & 82 \\ 92 & 92 & 92 & 92 & 92 & 92 \\ 95 & 95 & 95 & 95 & 95 & 95 \\ 93 & 93 & 93 & 93 & 93 & 93 \\ 83 & 83 & 83 & 83 & 83 & 83 \\ 94 & 94 & 94 & 94 & 94 & 94 \end{pmatrix}$ by transposing **shiftMat**. To compute $\Delta\mathbf{X}$, each row of **baseMat** can be thought of as a starting point and each row of **shiftMat** can be thought of as the end points with respect to the

corresponding starting point in **baseMat**.

2. Compute **shiftMat** − **baseMat** = $\begin{pmatrix} 0 & 10 & 13 & 11 & 1 & 12 \\ -10 & 0 & -3 & -1 & 9 & 2 \\ -13 & -3 & 0 & -2 & -12 & -1 \\ -11 & -1 & 2 & 0 & -10 & 1 \\ -1 & 9 & 12 & 10 & 0 & 11 \\ -12 & -2 & 1 & -1 & -11 & 0 \end{pmatrix}$.

3. Since all entries of the main diagonal of **shiftMat** − **baseMat** are always 0 and the lower triangular part of **shiftMat** − **baseMat** below the main diagonal contains the time-reversed changes in X, only the upper triangular part of **shiftMat** − **baseMat** above the main diagonal contains all possible time-oriented changes in **X**, i.e. $\Delta\mathbf{X}$[1]. Hence, for the **X** in the example,

$$\Delta\mathbf{X} = \begin{pmatrix} NaN & 10 & 13 & 11 & 1 & 12 \\ NaN & NaN & 3 & 1 & -9 & 2 \\ NaN & NaN & NaN & -2 & -12 & -1 \\ NaN & NaN & NaN & NaN & -10 & 1 \\ NaN & NaN & NaN & NaN & NaN & 11 \\ NaN & NaN & NaN & NaN & NaN & NaN \end{pmatrix}.$$

In general, for any set of time oriented sample values $\mathbf{X} = \begin{pmatrix} x_1 & x_2 & \ldots & x_N \end{pmatrix}$,

$$\Delta\mathbf{X} = \begin{pmatrix} NaN & x_2 - x_1 & \ldots & x_N - x_1 \\ NaN & NaN & \ldots & x_N - x_2 \\ \ldots & \ldots & \ldots & \ldots \\ NaN & NaN & \ldots & NaN \end{pmatrix}$$

[1] if ΔX_{min}, and ΔX_{max} in gradient search criteria were specified in percentage, $\Delta\mathbf{X}$ = (**shiftMat** − **baseMat**)./**baseMat**, with the main diagonal and the lower triangular part of the matrix filled with the special value NaN.

For the given $\mathbf{T} = \begin{pmatrix} 3 & 5 & 7 & 8 & 9 \end{pmatrix}$ in the example,

$$\Delta\mathbf{T} = \begin{pmatrix} NaN & 2 & 4 & 5 & 6 & 12 \\ NaN & NaN & 2 & 3 & 4 & 10 \\ NaN & NaN & NaN & 1 & 2 & 8 \\ NaN & NaN & NaN & NaN & 1 & 7 \\ NaN & NaN & NaN & NaN & NaN & 6 \\ NaN & NaN & NaN & NaN & NaN & NaN \end{pmatrix}$$

Step 2: Finding index pairs of $\Delta\mathbf{X}$ for which $-20 \leq \Delta x \leq 0$ yields

$$\mathbf{I}_{ij}^{\Delta x} = \left((2,5)^T, (3,4)^T, (3,5)^T, (3,6)^T, (4,5)^T\right)$$

where for each $(m,n)^T$in, $I_{ij}^{\Delta x}, m =$ row index, and $n =$ column index . By convention, rows in a matrix are indexed from top to bottom with the top row indexed as 1 and columns in a matrix are indexed from left to right with the left-most row indexed as 1.

Similarly, finding index pairs of $\Delta\mathbf{T}$ where $0 \leq \Delta t \leq 10$ yields

$$\mathbf{I}_{ij}^{\Delta t} = ((1,2)^T, (1,3)^T, (1,4)^T, (1,5)^T, (2,3)^T, (2,4)^T, (2,5)^T, (3,4)^T, (3,5)^T, (3,6)^T, (4,5)^T, (4,6)^T, (5,6)^T).$$

Finding the intersection of $\mathbf{I}_{ij}^{\Delta x}$ and $\mathbf{I}_{ij}^{\Delta t}$ yields

$$\mathbf{I}_{ij}^{hits} = \mathbf{I}_{ij}^{\Delta x} \cap \mathbf{I}_{ij}^{\Delta t} = \left((2,5)^T, (3,4)^T, (3,5)^T, (3,6)^T, (4,5)^T\right)$$

Since $\Delta X_{min} \leq \Delta x_{ij} \leq \Delta X_{max}$ and $\Delta T_{min} \leq \Delta t_{ij} \leq \Delta T_{max}$ for all indices (i,j) in $\mathbf{I}_{ij}^{hits}$, all changes in sample values from x_i to x_j within time periods t_i to t_j satisfy the constraint specified by the gradient search criterion. Hence, the gradient search criterion is satisfied for all periods from t_i to t_j for all pairs of (i,j) in $\mathbf{I}_{ij}^{hits}$. To represent the fact that the gradient search criterion was satisfied during the period from t_i to t_j on a boolean time line, one simply has to set all boolean values from t_i to t_j to 1. However, periods from t_i to t_j for all pairs of (i,j) in $\mathbf{I}_{ij}^{hits}$ may contain overlaps. Since converting overlapped periods into the boolean representation requires unnecessary computation, steps 3 and 4 of the gradient search algorithm (detailed below) are executed to minimize the duration of overlapped periods.

From each pair of indices, (i, j) in $\mathbf{I}_{ij}^{hits}$, one can derive a beginning time stamp t_i and an ending time stamp t_j that define a period $[t_i : t_j]$ during which a gradient search criterion is satisfied. Hence, i essentially defines the starting time stamp and j defines the ending time stamp of a period on a time line.

Step 3: Pairs of indices (i, j) in $\mathbf{I}_{ij}^{hits}$ with different i and identical j define periods with different starting time stamps but an identical ending time stamp. For $\mathbf{I}_{ij}^{hits}$ calculated in Step 2, three index pairs (2,5), (3,5) and (4,5) have a common column index $j = 5$. Since $[t_4 : t_5] \subset [t_3 : t_5] \subset [t_2 : t_5]$ on a time line, once the boolean values in the period $[t_2 : t_5]$ are set to 1, the boolean values in the periods $[t_4 : t_5]$, and $[t_3 : t_5]$ are set to 1 as well. In general, if $\mathbf{I}_{ij}^{hits}$ contains index pairs in which multiple row indices i are paired with the same column index j, only the index pair with the lowest row index i (representing the period with the earliest starting time stamp) should be retained. In the example, the index pairs (3,5) and (4,5) contain redundant information and should be discarded from $\mathbf{I}_{ij}^{hits}$. After completing step 3, $\mathbf{I}_{ij}^{hits} = \Big((2,5),(3,4),(3,6)\Big)$.

Step 4: Pairs of indices (i, j) in $\mathbf{I}_{ij}^{hits}$ with identical i but different j define periods with an identical starting time stamp but different ending time stamps. After Step 3, $\mathbf{I}_{ij}^{hits}$ contains 2 index pairs (3,4) and (3,6) with a common row index $i = 3$. Since $[t_3 : t_4] \subset [t_3 : t_6]$ on a time line, once the boolean values in the period $[t_3 : t_6]$ are set to 1, the boolean values in the period $[t_3 : t_4]$ are set to 1 as well. In general, if $\mathbf{I}_{ij}^{hits}$ contains index pairs in which multiple column indices j are paired with an identical row index i, only the index pair with the highest column index j (representing the period with the latest ending time stamp) should be retained. In the example, the index pair (3,4) contains redundant information and should be discarded from $\mathbf{I}_{ij}^{hits}$. After completing step 4, $\mathbf{I}_{ij}^{hits} = \Big((2,5),(3,6)\Big)$.

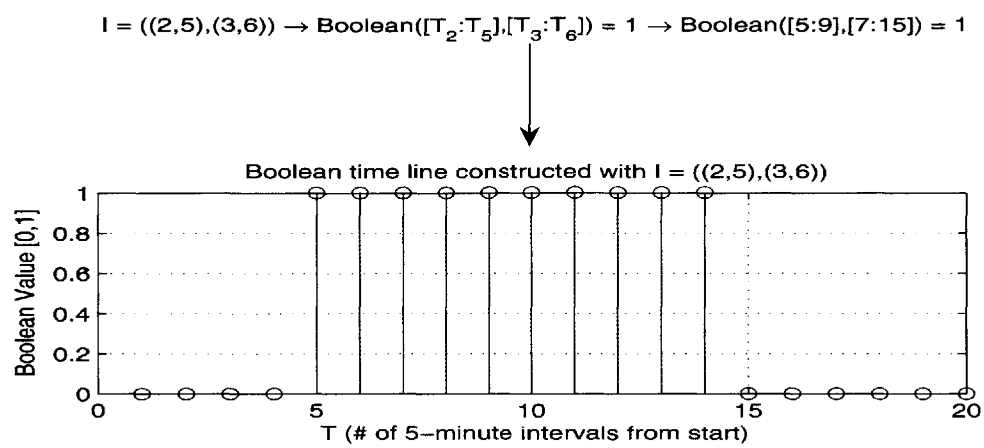

Figure 3-8: Graphical Illustration of how a boolean time line is constructed from $\mathbf{I}_{ij}^{hits}$.

Step 5: Figure 3-8 provides a graphical illustration of how a boolean time line is constructed from $\mathbf{I}_{ij}^{hits}$ in step 4. In the above example, it is assumed that the observed length of P is recorded as 20 in the *pidTimeLine* matrix. Hence, a boolean time line of length 20 is allocated with all its boolean values initialized to 0. Since $t_2 = 5, t_3 = 7, t_5 = 9$, and $t_6 = 15$, the boolean values for the periods $[5:9]$, and $[7:15]$ were set to 1. Since the periods $[5:9]$, and $[7:15]$ contain an overlapping period $[7:9]$, in Figure 3-8, it appears as though there was a single period $[5:15]$ during which the gradient search criterion was satisfied.

c) From a Boolean String to a Set of Onset, Offset Points

Figure 3-9 provides a graphical illustration of how a boolean time line can be converted into T_{on}, T_{off} representations. As shown in the figure, T_{on} stores a set of time stamps t_i that mark the beginning of a series of evenly sampled 1's on the boolean time line. For each t_i in T_{on}, there is a corresponding t_i in T_{off} that marks the end of the period that fulfils the time series search criteria.

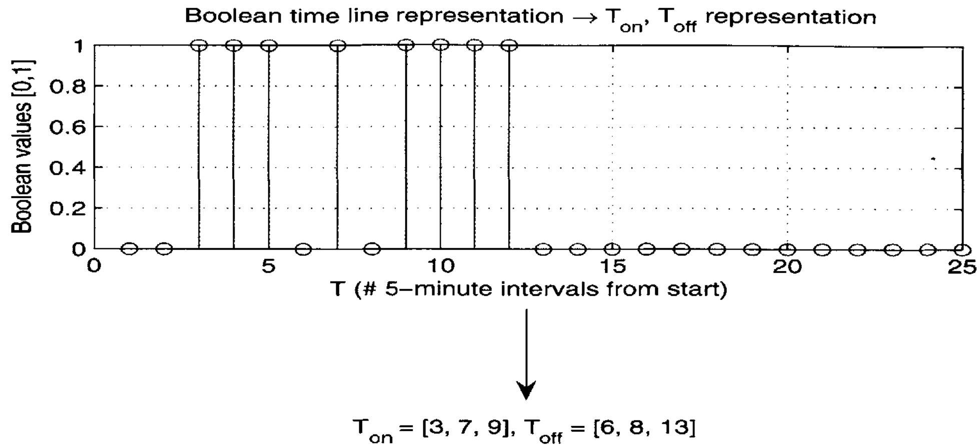

Figure 3-9: Graphical Illustration of how a boolean time line is converted into T_{on}, T_{off} representation.

3.6 Interface Design

The web-based interface for the MIMIC II search engine was jointly designed by Dr. Gari Clifford, Andrew Hung and Tin Htet Kyaw and was implemented by Andrew Hung.

3.6.1 Design Objectives

The interface for the search engine was designed with the following primary objectives:

1. The interface was designed to be fully compatible with the search engine. To ensure compatibility, the interface was implemented using MATLAB and MATLAB WebServer.

2. The interface was designed to be sufficiently user-friendly for clinicians without extensive knowledge about databases and query languages to use. Therefore, the interface incorporates many intuitive and user-friendly GUI (Graphical User Interface) features such as buttons, drop-down boxes and checkboxes to minimize the need for a user to type in their instructions.

3. The interface was designed to provide enough flexibility for sophisticated users with extensive database knowledge to specify complex queries composed of numerous basic query criteria joined together by logical operators.

4. The interface was designed to be extensively portable with minimum requirement for installation and use. The web-based implementation of the database allows users to access the search engine from any computer (a PC or a Mac running any standard operating systems such as Linux, Mac OS, or Windows) with an html-compatible web browser. The web-based interface also eliminates the need to install MATLAB in order to run the search engine.

5. The interface was designed to provide an automatic method to visualize the search results returned by the search engine without the use of a graphing program.

6. The interface was designed to serve as a bridge between the search engine and the Annotation Station [8], an open-source software system for visualizing and annotating clinical information in massive biomedical databases. The interface was designed to have the ability to convert the search results returned by the search engine into Annotation Station-compatible XML annotations. Since the comprehensive set of clinical information available for a patient can be loaded and visualized on the Annotation Station, having the ability to load the search engine's results as annotations enables the search results to be verified for correctness on the Annotation Station. Furthermore, researchers or clinicians using the Annotation Station to investigate patient records can use the search engine generated annotations to focus their investigations of the patient records.

3.6.2 Implementation

Figure 3-10 provides the flow chart of the web-based interface for the search engine. The flow of the program is explained below with a step-by-step execution of a sample query:

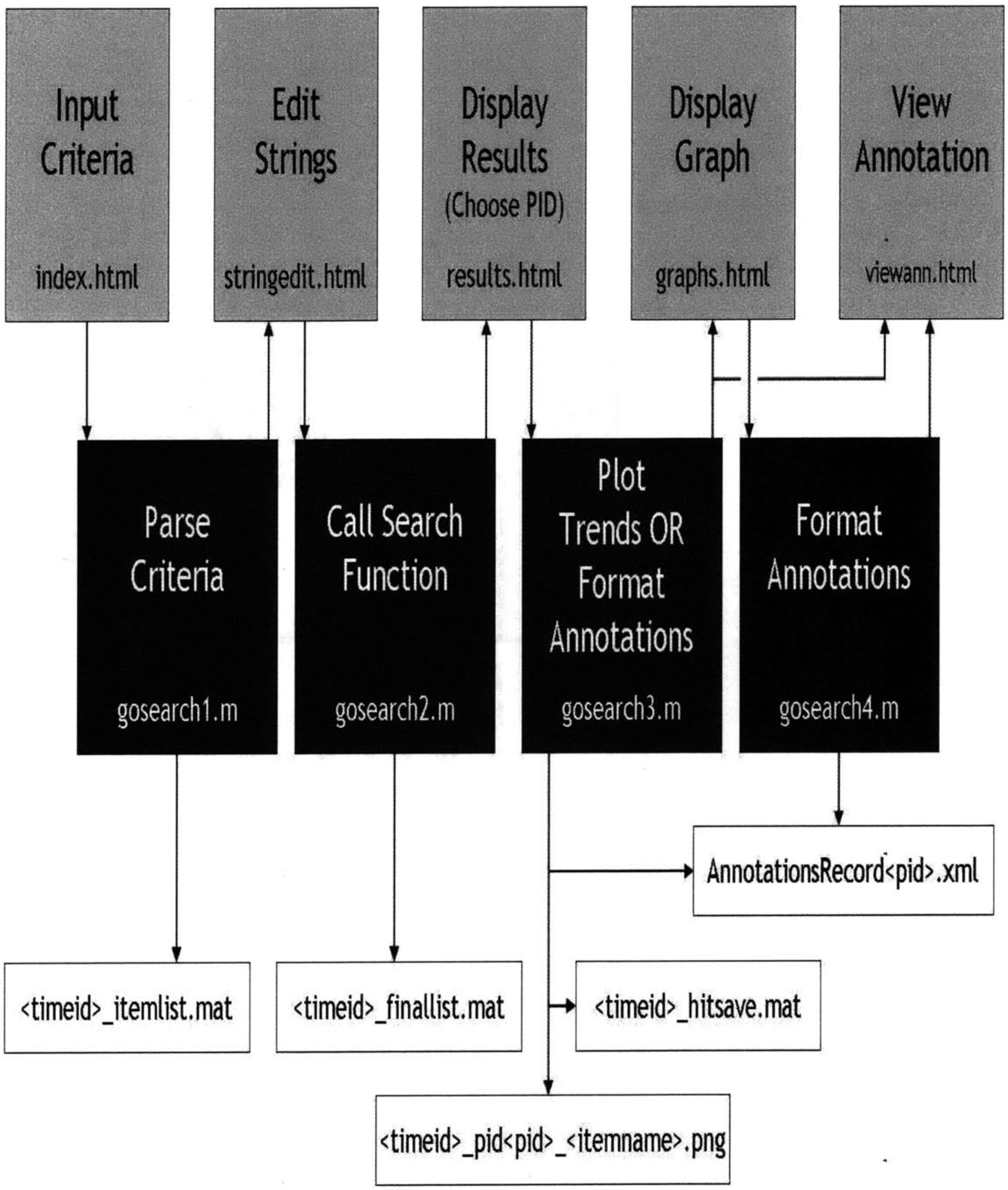

Figure 3-10: Flow chart of the web-based interface for the MIMIC II search engine. Grey boxes indicate html pages, black boxes MATLAB routines which collect parameters from the html pages and parse them into search-engine compatible query strings (*gosearch1.m*), execute the query using the search engine (*gosearch2.m*), plot trend graphs (*gosearch3.m*) or generate annotations in XML format (*gosearch4.m*). White boxes indicate relevant files: data(.mat), XML annotations(.xml) or figures(.png).

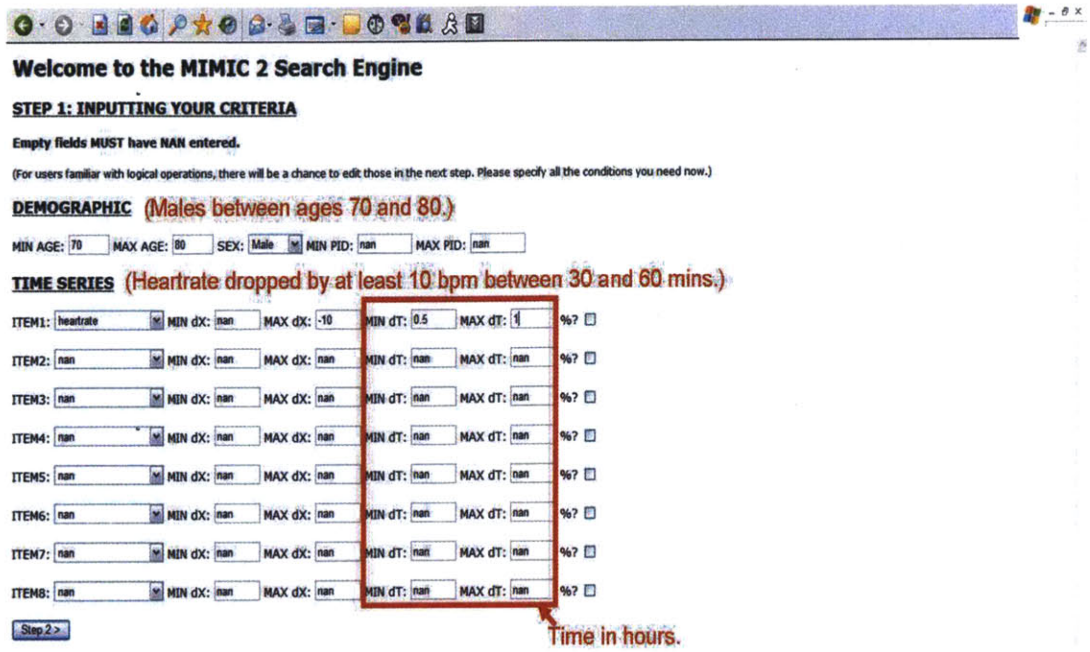

Figure 3-11: Step 1 of the web-based interface for the MIMIC II search engine. All entries in the page are initialized with NaN when the screen is launched. The demographic section allows demographic constraints to be specified for a search. The time series section allows up to 6 time series search criteria to be specified. The time series criteria are jointed together by the & operator. Clicking 'Step 2 >' passes the input from the page to *gosearch1.m* which in turn populates the html page for Step 2.

1. Figure 3-11 provides a screen shot of Step 1 of the web-based interface program for the search engine. All input boxes in the page are initialized with NaN at launch time. The row of input boxes on the web page in the demographic section allows a user to specify the constraints for a demographic query string. In Figure 3-11, the demographic constraints are set to find male patients with ages between 70 and 80.

   In the time series section of the page, there are 6 rows of inputs available to allow a user to specify up to 6 time series search criteria to compose a time series search string (discussed in Section 3.4.3.b). All the time series criteria specified are combined with the & operator. The dropdown boxes at the beginning of each row of input are filled with searchable items to allow a user to easily select an item to search. The checkbox at the end of a row of input needs to be checked if the values specified for dX_{min} and dX_{max} are to be interpreted as percentage values for a gradient search criterion. In Figure 3-11, only a single time series search criterion was set to find episodes in patient records where the heart rate dropped by at least 10 beats per minute within intervals that were between 30 and 60 minutes in length.

Once a user completes his or her specification of the demographic criteria and the time series search criteria by pressing the 'Step 2 >' button on the page, the MATLAB routine *gosearch1.m* is called which formats the input data from the page into search-engine compatible query strings and populate the html page for Step 2 wtih the query strings. When *gosearch1.m* is complete, the browser is redirected to the second page of the interface where the user-specified criteria are displayed as a demographic query string and a time series query string as required by the search engine.

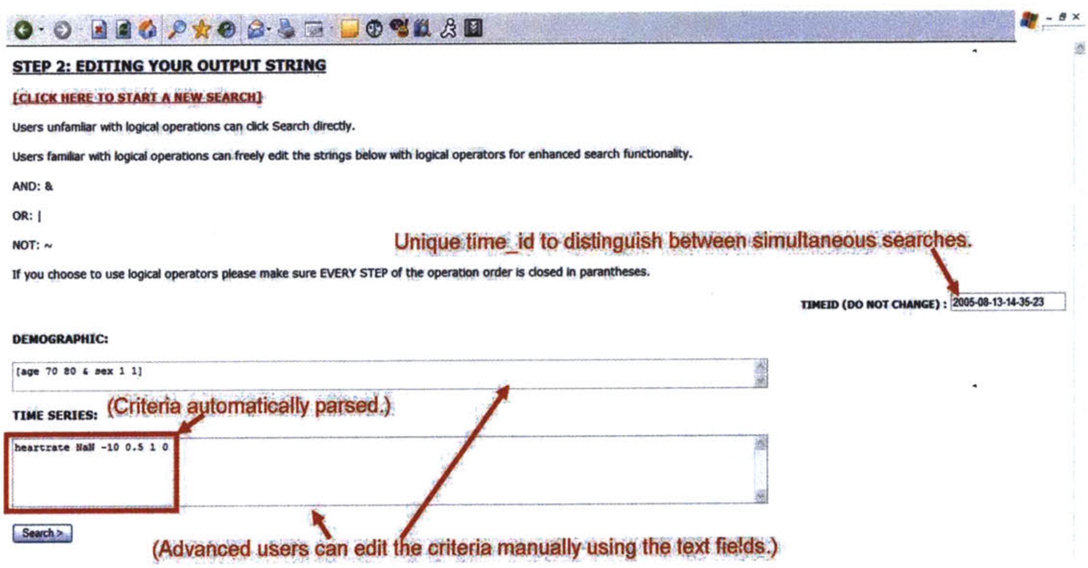

Figure 3-12: Step 2 of the web-based interface for the MIMIC II search engine. The demographic criteria and time series search criteria specified in Step 1 are formatted into the MIMIC II search engine compatible demographic query string and time series query string. A user is allowed to modify the query strings to specify more sophisticated queries. The TIMEID is a unique identifier generated for a query to facilitate the storage and identification of results generated for the query. Clicking 'Search >' passes the demographic and time series query strings to *gosearch2.m* which in turn passes them to *mimic2search.m* to execute the query.

2. Figure 3-12 provides a screen shot of Step 2 in the interface program. In this step, the demographic criteria and the time series criteria specified in Step 1 are formatted into search engine compatible demographic query string and time series query string which are displayed in modifiable text fields on the page. A user is allowed to modify or augment the query strings to specify more sophisticated queries. A unique TIMEID is also generated for the query to facilitate the storage and identification of results generated for the query. A user completes Step 2 by pressing the 'Search >' button at which point the interface program (*gosearch2.m*) issues a search to the MIMIC II search engine (*mimic2search.m*) providing as arguments the demographic query string and the time series query string collected from Step 2. There is also a link at the top of the page that allows a user to return to Step 1.

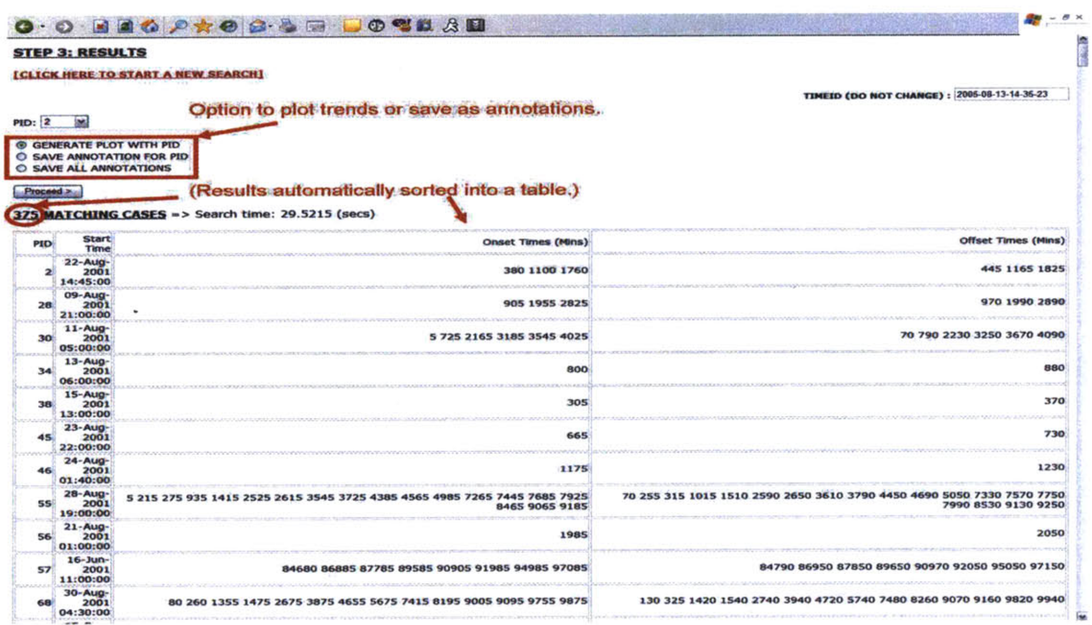

Figure 3-13: Step 3 of the web-based interface for the MIMIC II search engine. The search results returned by the MIMIC II search engine, together with the time it took for the search, are displayed on this page. The page also provides 3 options: 1) to generate plots of a patient record, 2) to save the search result for a particular record as an annotation and 3) to save all search results as annotations.

3. Figure 3-13 provides a screen shot of Step 3 in the interface program. Specifically, the figure shows a snapshot of the search results returned by the search

engine for the search criteria specified in Figure 3-11. The number of cases (patient records) that match the criteria, and the time it took for the search engine to perform the search are clearly displayed. Furthermore, all the search results are sorted into a table with 4 columns displaying for each patient record the pid, the start time, the onset time and offset time (in minutes from the start time). The pids of all patient records returned by the search engine are also listed in the dropdown box near the top of the page. Just as in Step 2, there is also a link at the top of the page that allows a user to return to Step 1 to begin a new search.

There are 3 options provided for a user to proceed from Step 3:

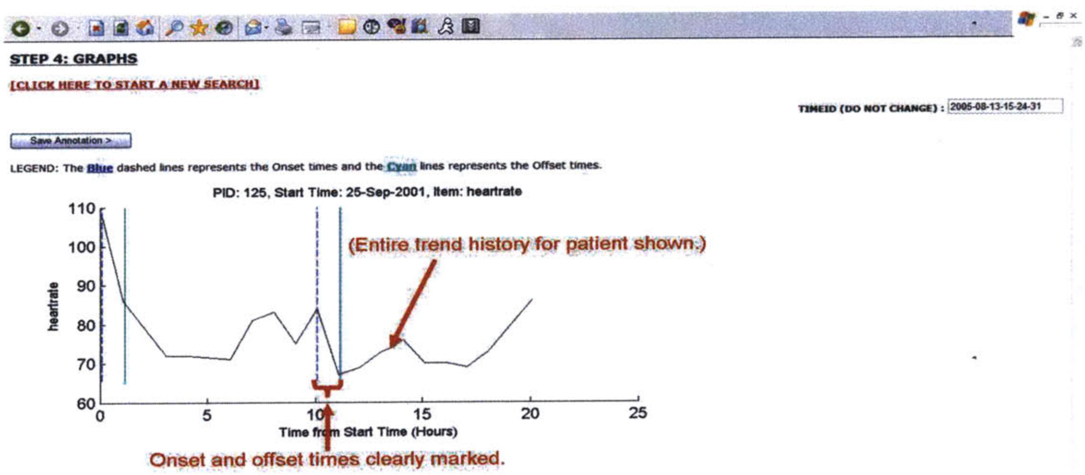

Figure 3-14: Screen shot of a trend plot generated for a patient record returned as a hit by the MIMIC II search engine. The plots are generated by a function call to *gosearch4.m*. In the figure, the entire trend record of heart rate is displayed. The onset time stamps are marked with dashed blue vertical lines whereas the offset time stamps are marked with solid cyan vertical lines.

(a) As the first option, a user can generate plots to visualize the search results returned for the patient record with the pid selected in the drop-down box. The plots are generated by a function call to *gosearch4.m*. An example of such a plot is shown in Figure 3-14. In the figure, the entire trend record of heart rate is displayed. The onset time stamps are marked with dashed blue vertical lines whereas the offset time stamps are marked with solid cyan vertical lines. Only 1 plot for the heart rate trend was generated because there was only a single time series criterion defined based on the heart rate. If there were multiple time series criteria specified in a query, there would be multiple plots generated: one for each criterion. A user can also save the search result of the patient record displayed on the page as an annotation by pressing the 'Save Annotation >' button at the top of the page. If a user chooses this option, an XML annotation of the patient record will be generated and the user will be redirected to the directory where the annotation is stored as discussed in b).

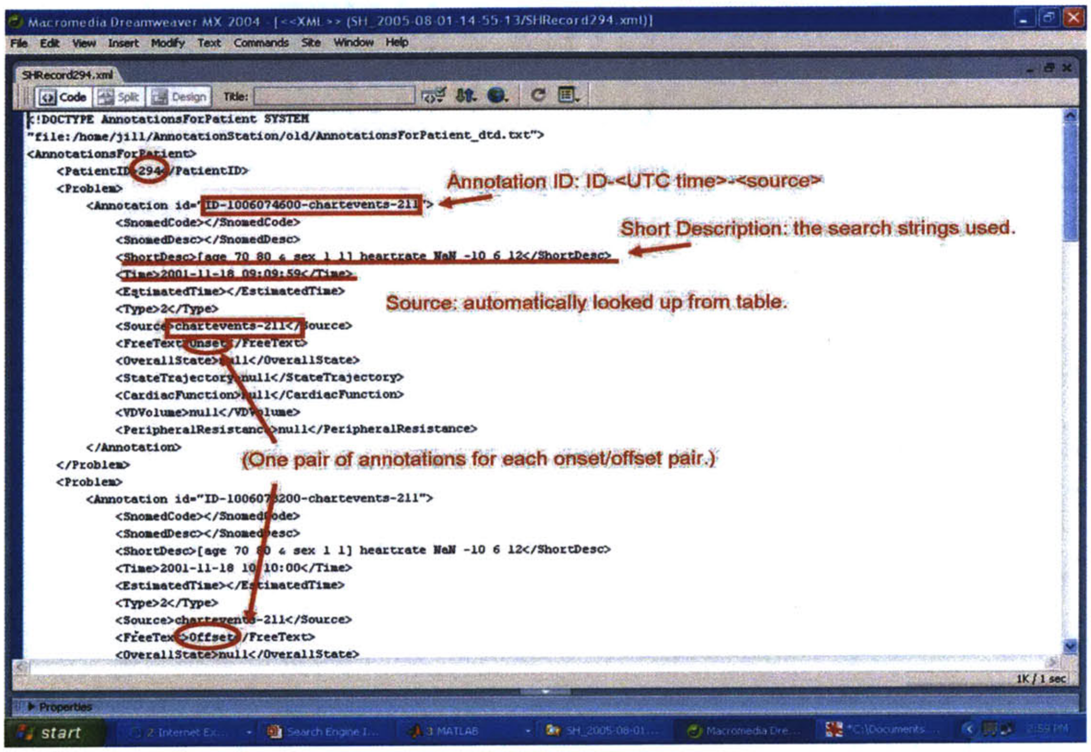

Figure 3-15: A sample annotation generated from a hit returned by the MIMIC II search engine.

(b) As the second option, a user can also save the search results returned for the patient record with the pid selected in the drop-down box. If a user selects to proceed with this option, the MATLAB routine *gosearch4.m* is called to convert the search result into an XML annotation [8] in a special directory on the server running the MIMIC II search engine. A sample

XML annotation is shown in Figure 3-15. A zip file containing the XML annotation is also created in the same directory. The directory name was generated based on the time stamp at which the query was executed. After the annotation file and the zip file are successfully generated, the user is redirected to the directory (as shown in Figure 3-16) so that he or she can download the files to a local storage disk.

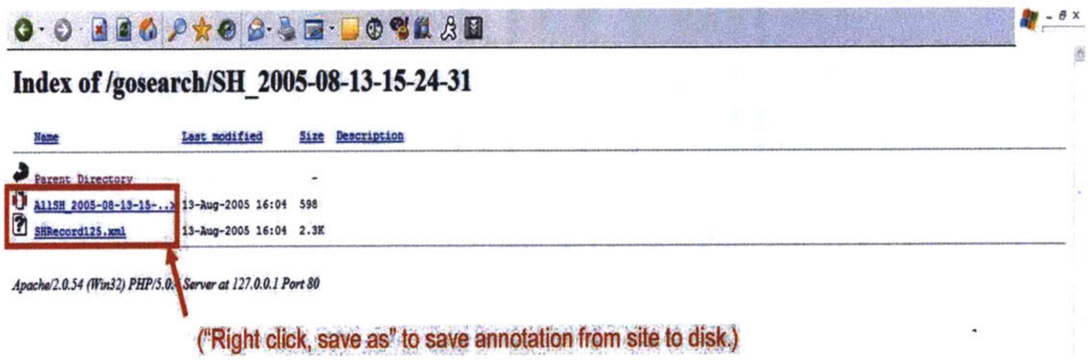

Figure 3-16: Screen shot of the content of a directory where the search engine generated annotations are saved. The XML annotation can be loaded on the Annotation Station to visualize the search results. The zip file is a compressed archive of all the annotations in the directory that the user has requested be generated. The directory name was generated based on the time stamp at which the query was executed.

(c) As the third option, a user can save all patient records returned as hits by the search engine, each as an XML annotation. If this option is selected, each patient record returned by the search engine is saved as a separate XML annotation (Figure 3-15), and stored in the special directory created to save annotations for the query. A zip file containing all XML annotations is also created in the same directory to facilitate downloading. After all annotation files and the zip file are successfully created, the user is redirected to the directory where all those files are stored (Figure 3-16).

3.6.3 Strengths and Limitations

Strengths

Based on the experience of using the interface, the implementation of the search engine interface has the following strengths:

- The interface works seamlessly with the search engine. Yet, the implementation of the interface is cleanly decoupled from that of the search engine. In fact, there is only a single function call from the interface to the search engine that links the 2 systems together.

- The interface not only provides an intuitive way for people without extensive knowledge about databases or MATLAB to perform time series searches but also provides flexibility for sophisticated users to execute complex queries.

- The interface is portable: it has been tested on PCs running Windows XP as well as Linux operating systems.

- The interface provides an easy method to visualize the results returned by the search engine.

- The interface bridges the gap between the search engine and the Annotation Station through the generation of XML annotation files.

Limitations

The search engine has also been observed to have the following limitations due to its dependence on the MATLAB WebServer:

- Canceling or aborting an issued query is not possible in the current implementation. Once a query is issued to the MATLAB WebServer, there is no way to terminate it prematurely.

- Overloading the WebServer with multiple requests from various sources is known to cause the program to crash. This may have been caused by the limited computation resources available on the server. Therefore, one may need to run the search engine server on a computer with abundant memory and a fast processor. Alternatively, efficient queuing or parallelizing the searches may also solve this problem of resource overuse.

- There has been significant memory leakage detected with the MATLAB WebServer running overnight under Windows XP to support the search engine interface. Although this may be an operating system specific problem, one can alternatively explore ways to decouple the dependence of the search engine interface on the MATLAB WebServer.

3.7 Performance, Case Studies and Discussion

The biggest challenge in testing and evaluating the performance of a search engine for a clinical information database such as MIMIC II lies in the challenge of quantifying missed detections for time-series searches. For a search engine that performs time series searches, the following 2 types of missed detections are possible:

- The search engine might detect some episodes or periods in a record during which the time series criteria were satisfied while missing other episodes during which the criteria were satisfied. Such a miss is known as an *episode miss.*

- The search engine might fail to detect all episodes or periods in a record during which the time series criteria of a query were satisfied and fail to identify the record as a hit. Such a miss is known as a *record miss.*

One possible way to find missed detections by a search engine is to manually perform the searches on patient records by visually evaluating the plots of sample values for the records. However, visually evaluating the plot is not only extremely laborious and time-consuming but also unreliable at uncovering all the missed detection. On the other hand, an automatic verification algorithm for a search engine requires a pre-annotated database in which all episodes of hits are identified by machine-readable annotations. Since annotating the MIMIC II database is still an ongoing effort, a comprehensive evaluation of the missed detections of the search engine for the MIMIC II database needs to be deferred until the database is fully annotated.

A summary of the tests conducted to evaluate the performance of the search engine and the results from those tests are described in Section 3.7.1. Case studies on a selected set of patient records returned as hits from the tests are then provided in Section 3.7.2. The limitations of the current search engine are highlighted and then possible ways of overcoming those limitations are suggested in the case studies.

3.7.1 Performance Tests and Results Summary

Table 3.9 provides a summary of the tests conducted to evaluate the performance of the search engine on the MIMIC II database. The results returned by the search engine for the searches listed in Table 3.9 are summarized in Table 3.10.

Speed and Scalability

As described in Section 3.5.1, to conduct a gradient search of an item for a patient record, the search engine calculates all possible gradient values from the samples and then find the gradient values that meet the constraint specified in the criterion. To keep the algorithm simple and to minimize storage overhead, no gradient values were precomputed. In return for a relatively simple gradient search algorithm with minimum storage overhead, the algorithm is required to do extensive amount for computation to perform a gradient search. In fact, the order of growth for the amount of computation required by the algorithm is $O(N^2)$ where, N is the number of sample values of the item of interest in the patient record. As evidenced in Table 3.10, queries that include gradient searches on items with relatively high numbers of samples such as heart rate and mean arterial blood pressure (Cases 4 and 8) take significantly more time to complete compared to other searches.

The order of growth in the amount of memory required to perform a gradient search is also $O(N^2)$, where N is the number of sample values. Hence, for high

Table 3.9: Summary of tests conducted to evaluate the performance of the search engine on the MIMIC II database.

Test Case	Search Objective	Evidence Searched	Time Series Query String
1	To find evidence of metabolic acidosis	$pH \leq 7.2$ AND $pCO_2 \leq 35$ AND $Lactate \geq 2.5$	((ph NaN 7.2 NaN NaN 0 & paco2 NaN 35 NaN NaN 0) & lactate 2.5 NaN NaN NaN 0)
2	To find evidence of acute renal failure	$Creatinine \geq 2$ AND $\Delta Creatinine \geq 200\%$ within $\Delta T \leq 48hours$	(creatinine 2 NaN NaN NaN 0 & creatinine 200 NaN NaN 48 1)
3	To find evidence of multi-organ failure	$Creatinine \geq 1.5$ AND $\Delta Creatinine \geq 100\%$ within $\Delta T \leq 48hours$; AND $AST \geq 40$ AND $\Delta AST \geq 50\%$ within $\Delta T \leq 48hours$; AND $ALT \geq 40$ AND $\Delta AST \geq 50\%$ within $\Delta T \leq 48hours$.	(((((creatinine 1.5 NaN NaN NaN 0 & creatinine 100 NaN NaN 48 1) & ast 40 NaN NaN NaN 0) & ast 50 NaN NaN 48 1) & alt 40 NaN NaN NaN 0) & alt 50 NaN NaN 48 1)
4	To find evidence of hemorrhagic shock	$\Delta HR \geq 50\%$ AND $\Delta ABP_{mean} \leq -20\%$ within $\Delta T \leq 6hours$	(heartrate 50 NaN NaN 6 1 & artbpmean NaN -20 NaN 6 1)
5	To find times at which a vasoconstrictor is started or increased significantly	$\Delta Levophed \geq 100\%$ within $\Delta T \leq 0.5hour$	levophed 100 NaN NaN 0.5 1
6	To find evidence of myocardial infarction	$Troponin \geq 0.1$	troponin 0.1 NaN NaN NaN 0
7	To find evidence of intubation (mechanical ventilation onset)	$TidalVolumeSet \geq 0.01$	tidalvolumeset 0.01 NaN NaN NaN 0
8	To find evidence of probable paroxysmal tachyarrhythmia	$\Delta HR \geq 40bpm$ within $\Delta T \leq \frac{1}{12}hour$	heartrate 40 NaN NaN $\frac{1}{12}$ 0

Table 3.10: Summary of Results for tests in Table 3.9.

Test Case	Search Objective	Search Time (seconds)	Number of Patient Records Returned as Hits
1	To find evidence of metabolic acidosis	1.442	28
2	To find evidence of acute renal failure	0.831	13
3	To find evidence of multi-organ failure	1.472	3
4	To find evidence of hemorrhagic shock	118.921	526
5	To find times at which a vasoconstrictor is started or increased significantly	0.992	33
6	To find evidence of myocardial infarction	1.052	187
7	To find evidence of intubation (mechanical ventilation onset)	2.523	1119
8	To find evidence of probable paroxysmal tachyarrhythmia	62.910	124

resolution signals with many samples, it may not be possible to perform gradient searches using the algorithm implemented in the search engine.

To improve the speed of the search engine, methods to precompute, store and index gradient information efficiently should be explored for faster gradient searches. To improve the scalability of the gradient search algorithm, algorithms that minimize the number of gradient computations should be explored.

3.7.2 Case Studies and Discussion

The following case studies are presented to illustrate the strengths and weaknesses of the search engine on the MIMIC II database. All 3 cases were returned as hits by the search engine to test cases 1 – 3 in Table 3.9. To evaluate the accuracy of the search engine in identifying those cases as hits, the patient record for each case was carefully inspected by investigating relevant clinical information of the patient (including trend plots, progress reports and discharge summaries) on the Annotation Station [8] with the help of a clinician.

Case Study 1: A Hit to the Search for Metabolic Acidosis

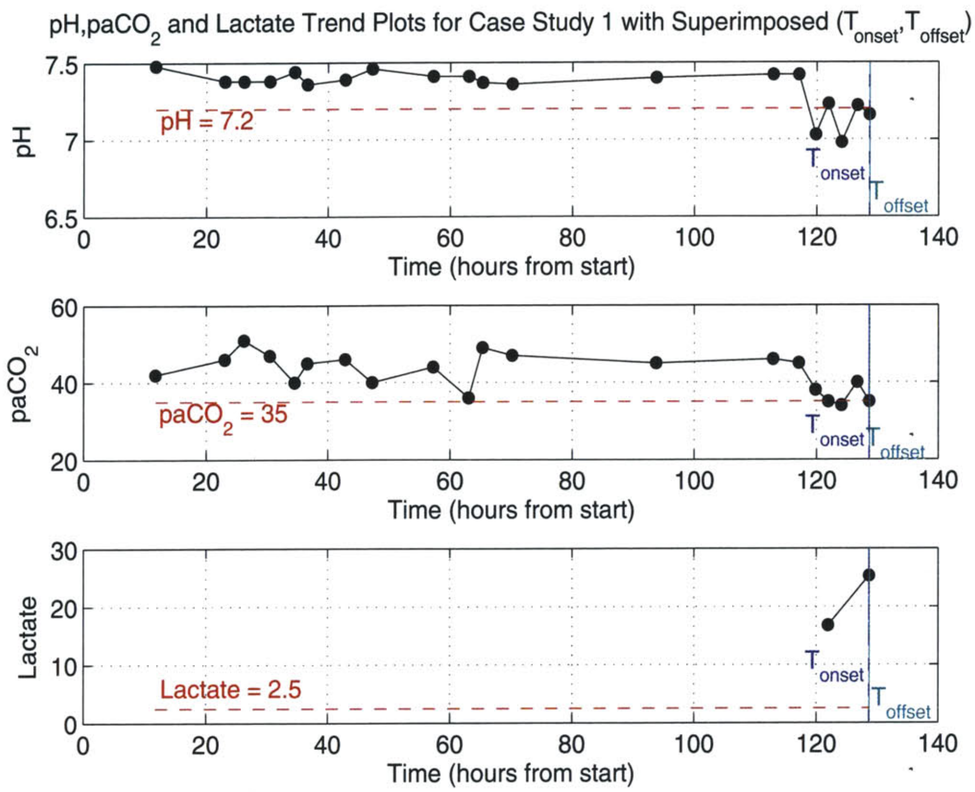

Figure 3-17: pH, $paCO_2$ and Lactate Trend Plots for Case Study 1.

A patient record (pid = 39) returned as a hit to the search for metabolic acidosis is presented for this case study. The relevant trend plots for the patient record with superimposed (T_{onset}, T_{offset}) are presented in Figure 3-17. As evident in the trend plots in Figure 3-17, the search engine successfully detected the single time point at which all relevant sample values satisfy the criteria for metabolic acidosis (specified in Table 3.9).

Investigating the trend plots of the patient record on the Annotation Station (Figure 3-18) shows that the patient indeed suffered from metabolic acidosis near the end of the record. An examination of the patient's discharge summary (Appendix B.1) also confirms the evidence of metabolic acidosis towards the end of the patient's stay in the ICU. Therefore, the search engine succeeded in identifying an episode of metabolic acidosis for this particular patient.

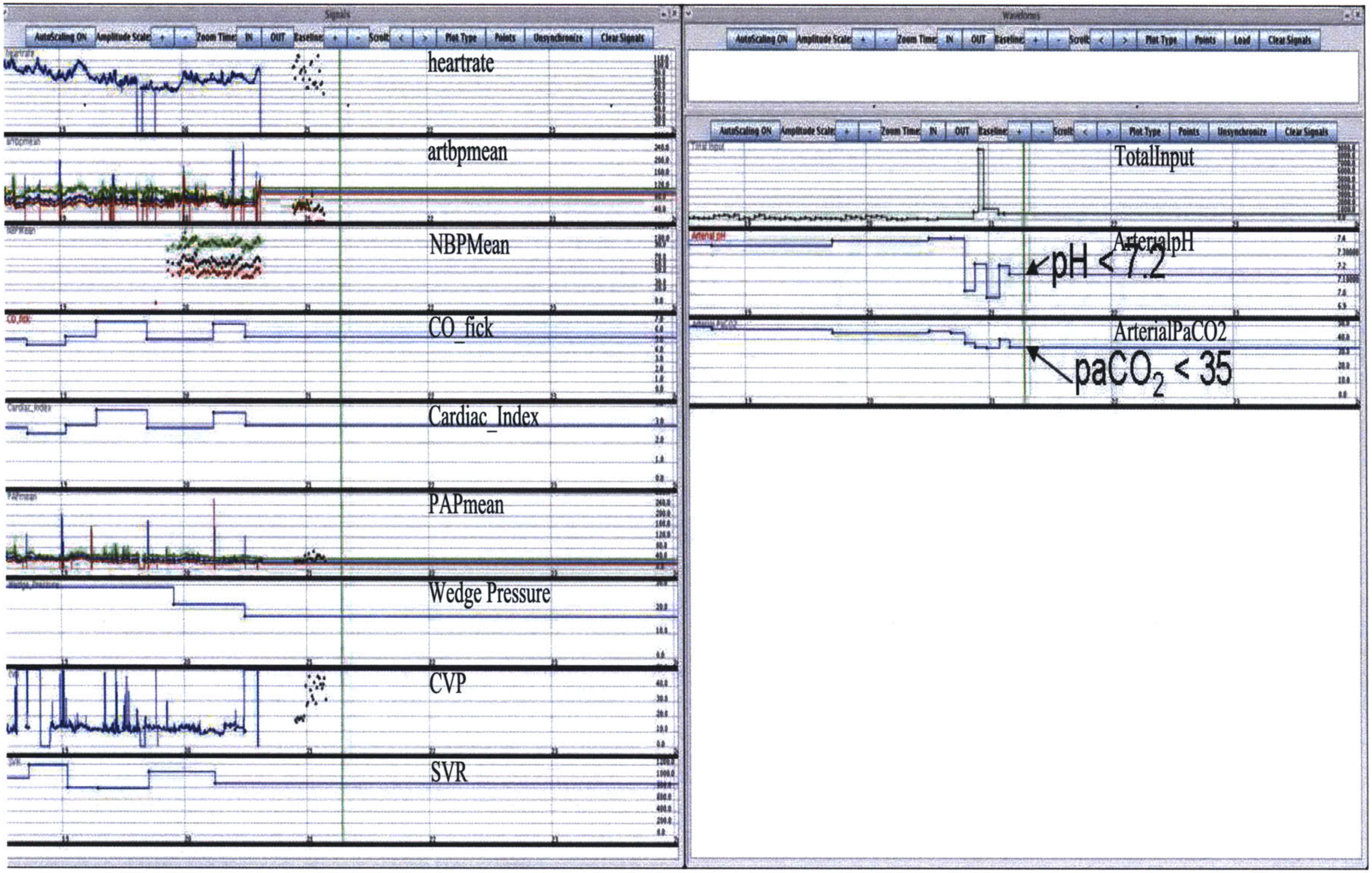

Figure 3-18: Relevant Trend Plots for Case Study 1 from the Annotation Station.

Case Study 2: A Hit to the Search for Acute Renal Failure

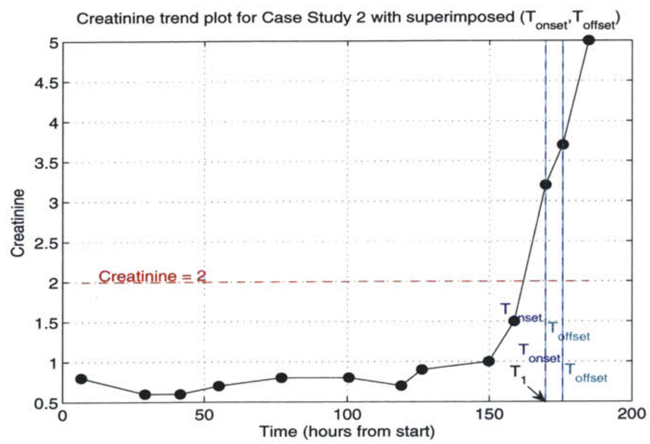

Figure 3-19: Creatinine Trend Plot for Case Study 2.

A patient record (pid = 67) returned as a hit to the search for acute renal failure is presented for this case study. The creatinine trend plot for the patient record with superimposed (T_{onset}, T_{offset}) is presented in Figure 3-19. As evident in the creatinine trend plot, the search engine successfully detected 2 time points at which the corresponding creatinine sample values satisfy the criteria for acute renal failure (listed in Table 3.9). Examining the trend plots of the patient record on the Annotation Station (Figure 3-20) and the patient's discharge summary (Appendix B.2) confirmed that the patient indeed suffered from acute renal failure towards the end of his stay in the ICU. Therefore, the search engine was successful at identifying an episode of acute renal failure for this patient.

Virtual Segmentation of a Physiological Episode: By visually inspecting Figure 3-19, one could argue that the period from T_1 to the end of the record should be identified as a single period during which the patient suffered from acute renal failure. The search engine identified the period as 2 separate time points because the search engine, as it is implemented, does not interpolate any sample values between irregularly sampled measurements. To avoid such virtual segmentation of physiological episode, an interpolation scheme needs to be implemented for the search engine.

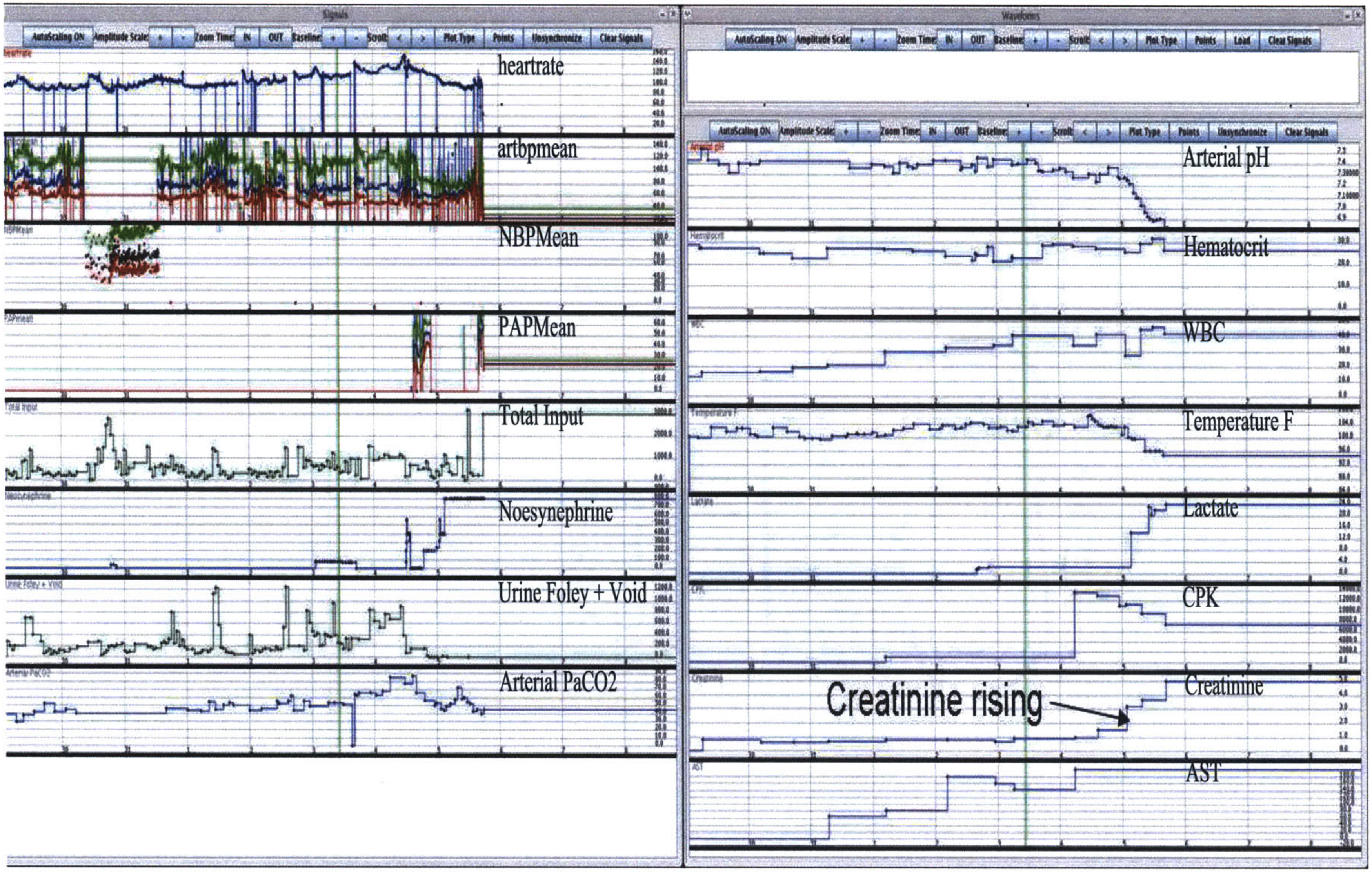

Figure 3-20: Relevant Trend Plots for Case Study 2 from the Annotation Station.

Case Study 3: A Hit to the Search for Multi-organ Failure

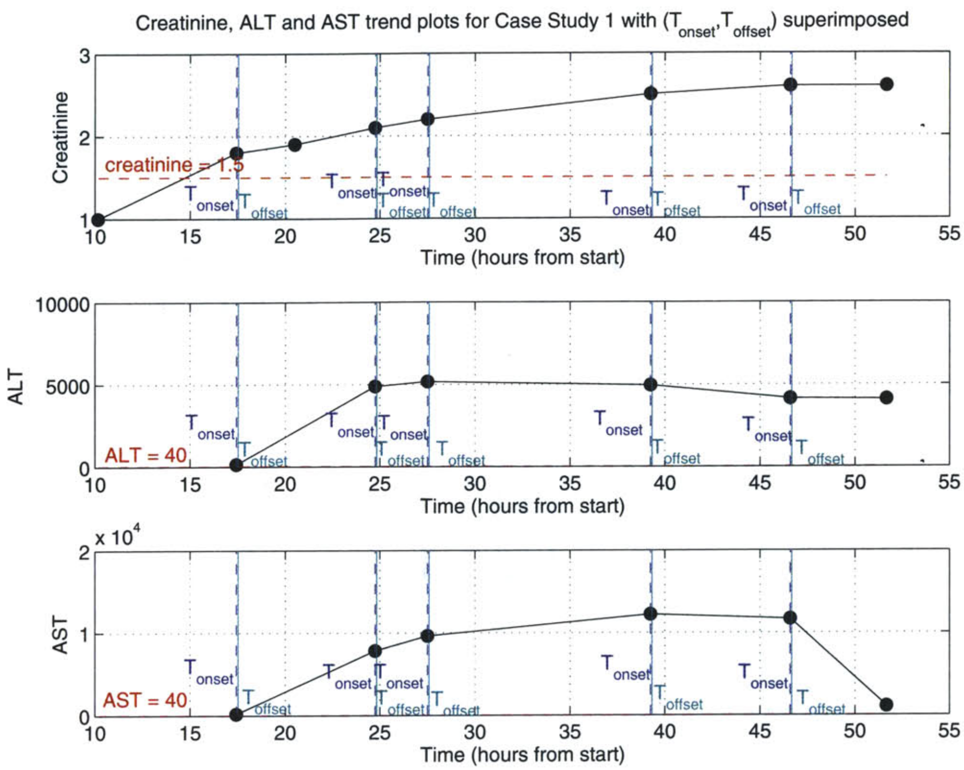

Figure 3-21: Creatinine, ALT and AST Trend Plot for Case Study 3.

A patient record (pid = 9320) returned as a hit to the search for multi-organ failure is presented for this case study. The creatinine, ALT and AST trend plot for the patient record with superimposed (T_{onset}, T_{offset}) is presented in Figure 3-21. As evident in the trend plots, the search engine successfully detected 5 time points at which the corresponding sample values satisfy the criteria for multi-organ failure (listed in Table 3.9). Examining the trend plots of the patient record on the Annotation Station (Figure 3-22) and the patient's discharge summary (Appendix B.3) confirmed that the patient indeed suffered from multiple organ failures due to cardiogenic shock towards the end of his stay in the ICU. Therefore, the search engine was successful at identifying an episode of multi-organ failure for this patient.

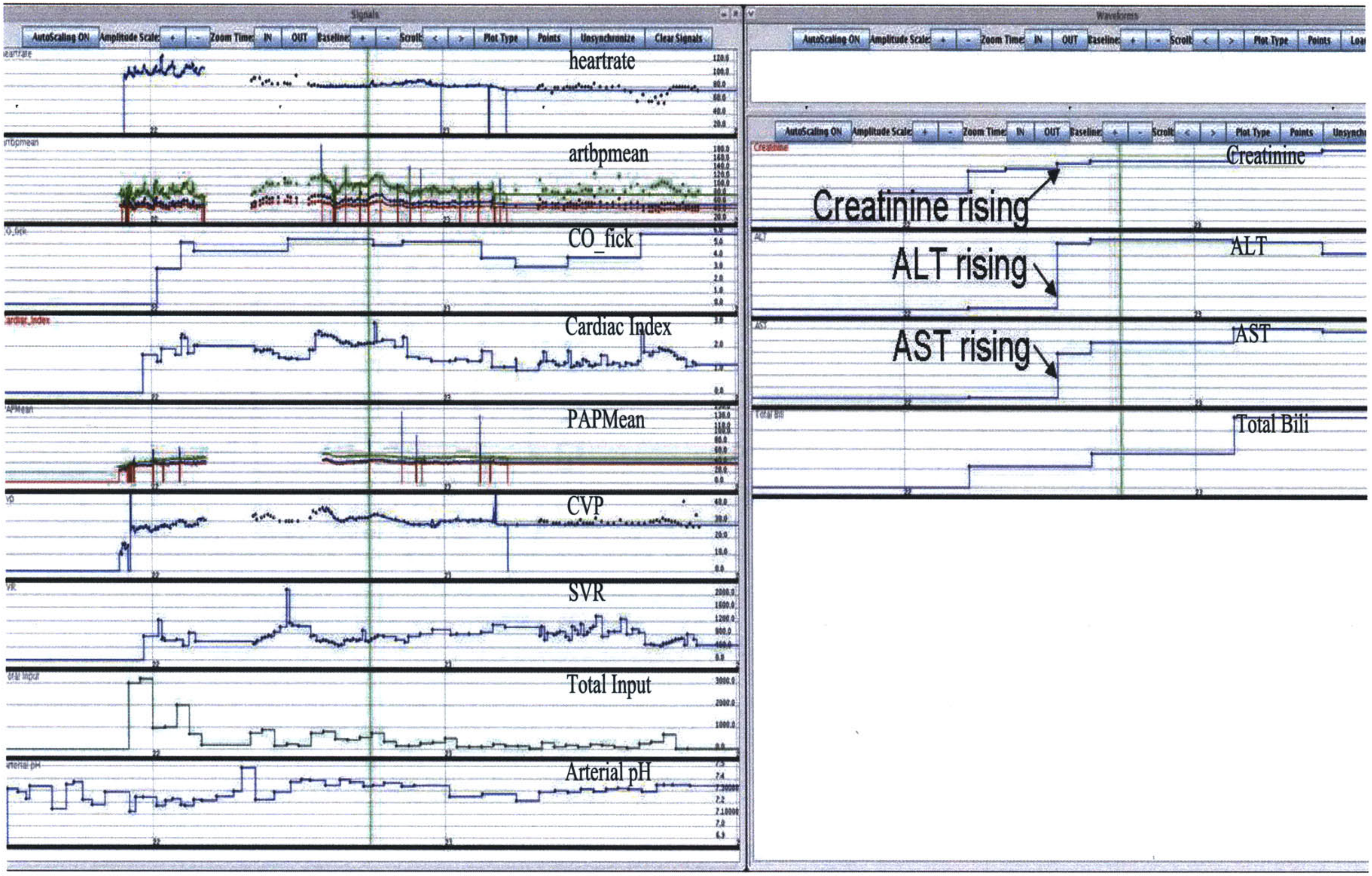

Figure 3-22: Relevant Trend Plots for Case Study 3 from the Annotation Station.

Detecting hits when measurements are not sampled synchronously: The search engine was able to detect multi-organ failure for this patient record because measurements of creatinine, ALT and AST were sampled at the same time when available. If the measurements were not sampled at the exact same time, the search engine would have failed to detect the episode of multi-organ failure for this patient.

However, it is still possible to detect multi-organ failure using the current search engine implementation when the measurements are not sampled at the same time. One possible way is to first perform 3 different queries with criteria based on creatinine, ALT and AST respectively. For each patient record that is a hit in all 3 queries, one could generate trend plots for creatinine, ALT and AST with the (T_{onset}, T_{offset}) returned from each query superimposed on the trend plots. Given the trend plots, one could identify episodes of multi-organ failure based on the visual evidence provided by the trend plots.

Alternatively, it is also possible to develop algorithms to automatically detect hits when measurements are not synchronously sampled. The following are a few suggestions for future implementation:

- The most straightforward method is to implement an algorithm that can efficiently interpolate unevenly sampled measurements.
- Alternatively, one could develop an algorithm to record the 'near-misses', in which the criteria for different items were not satisfied at the exact same time but within a time window of finite length. Once the 'near-misses' are identified, a set of subjective criteria can be used to categorize those 'near-misses' as either hits or misses.
- The current implementation of the search engine uses a boolean time line to represent and combine results for time series searches. Hence for the result at each time stamp could only be either a 'hit' or a 'miss'. As a third alternative, one could explore using fuzzy-logic [30] instead of binary boolean logic to represent results for time series searches.

Chapter 4

Conclusions and Suggested Future Work

The first half of this thesis describes the effort in converting the trend and waveform data from the proprietary Philips format to the open-source WFDB format. The second half of this thesis describes the design and implementation of a simple search engine that is capable of performing time series searches on the clinical information in the Carevue repository in the MIMIC-II database.

This chapter provides a brief summary of the 2 projects described in this thesis and suggests directions for future work.

4.1 Summary

In Chapter 2, the proprietary data schema for the trend and waveform data, along with the problems associated with each schema, were described in detail. The data conversion algorithms developed to solve the problems of the proprietary data schema and convert the data into the open-source WFDB format were also presented in this chapter. The remaining problems regarding the trend and waveform data were discussed and steps to improve the data collection effort were suggested.

Chapter 3 presented the design and implementation of a simple search engine for time series searches on clinical information using simple algorithms with little storage overhead. The design of the data structures, algorithms and interface for the search engine was discussed in detail. Results from time series searches to detect physiological events of clinical interest were presented. Detailed case studies on 3 patient records returned as hits for queries to detect clinically interesting physiological events were performed to highlight the strengths and limitations of the search engine.

4.2 Suggested Future Work

Data conversion algorithms for future proprietary format: The manufacturer has been designing a new and potentially improved proprietary format for trend and waveform data. Hence, the data conversion algorithms presented in this thesis need to be modified to convert trend and waveform data in the new proprietary format into the open-source WFDB format.

Algorithms for the search engine to detect physiological events when measurements are not synchronously sampled: As discussed in the case studies in Section 3.7.2, the search engine may fail to detect physiological events when clinical measurements were not synchronously sampled. Some possible methods to overcome this limitation of the search engine were also presented in Section 3.7.2.

Algorithms to improve the speed and scalability of the search engine: As presented in Section 3.7.1, the gradient search algorithm in the search engine has $O(N^2)$ growth rate in the number of computation cycles and memory usage, (N is the number of samples of an item in a patient record). Therefore, to improve the speed and scalability of the search engine, alternative methods to calculate and search for gradients need to be explored.

Algorithms to eliminate noise in data: Clinical information in an ICU are prone to various types of noise such as signal interference, measurement errors and device malfunctions. The current implementation of the search engine performs searches on only human-verified measurements and other data that are minimally prone to noise such as lab results and medications. However, the data is still susceptible to random noises which may lead the search engine to produce false hits. Therefore, in order to minimize the number of false hits the search engine can produce, it may be necessary to develop algorithms that can detect and eliminate noise and artifacts in data while performing searches.

Algorithms to perform time series searches on trend and waveform data: The bulk of information in a clinical information database is made up of high-resolution trend and waveform signals. Those trend and waveform signals not only provide a wealth of information regarding a patient's physiological state but also impose significant challenges in computational overhead for a search engine. Furthermore, the trend and waveform signals represent clinical data that are most susceptible to noise. However, to detect physiological episodes of clinical interest with high reliability, one needs to investigate the evidence found only in the trend and waveform signals. Therefore, it is necessary to develop algorithms that can perform time series searches on high-resolution physiological signals to have a complete search engine for a clinical information database.

Decouple the dependency of the search engine interface on the MATLAB WebServer: As discussed in Section 3.6.3, the MATLAB WebServer imposes strict limitations on the functionality of the web-based interface developed for the search engine. The dependence of the interface on the MATLAB WebServer should be minimized in order to improve the performance of the interface.

Optimize the search engine performance by implementing the algorithms in a compiled language: Programs written in MATLAB are interpreted 'on the fly' and thus are considerably slower than those written in compiled languages such as the C programming language. Hence, to improve the performance of the search engine, one could implement the search engine in a compiled language.

Optimize the search engine performance by parallelizing searches: Parallelizing time series queries on different patient records can significantly improve the performance of the search engine.

Appendix A

Full list of content in *columnMappings*

Table A.1: Content of *columnMappings* array.

Cell	Item	Description
1	pid	patient identification number
2	T	time elapsed from the start of patient record *(in units of 5 minutes)*
3	cvp	central venous pressure
4	heartrate	heart rate
5	artbpsys	arterial blood pressure *(systolic)*
6	artbpdias	arterial blood pressure *(diastolic)*
7	artbpmean	arterial blood pressure *(mean)*
8	PAPmean	pulmonary artery pressure *(mean)*
9	cofick	Fick cardiac output
10	cotherm	thermodilution cardiac output
11	iabp	intra-aortic balloon pump
12	ci	cardiac index
13	svr	sustained virologic response
14	strokevolume	stroke volume
15	pcwp	pulmonary capillary wedge pressure
16	resp	respiratory rate
17	spo2	Saturation Périphérique en Oxygène *(Pulse Oximetry)*
18	fio2set	fraction of inspired oxygen
19	minvolume	minimal respiratory volume
20	o2flow	oxygen flow

continued on the next page

Table A.1 – (*continued*)

Cell	Item	Description
21	peepset	positive end expiratory pressure *(set)*
22	peakinspres	peak inspiratory pressure
23	respratespont	respiratory rate *(spontaneous)*
24	respratetotal	respiratory rate *(total)*
25	resprateset	respiratory rate *(set)*
26	tidalvolumeobserved	tidal volume *(observed)*
27	tidalvolumeset	tidal volume *(set)*
28	tidalvolumespont	tidal volume *(spontaneous)*
29	sao2	oxygen saturation of arterial blood
30	waveformvent	ventilator waveform
31	icp	intracranial pressure
32	glascow	Glascow Coma scale
33	baseexcess	base excess
34	artco2	arterial carbon dioxide
35	paco2	partial pressure of arterial carbon dioxide
36	pao2	partial pressure of arterial oxygen
37	pH	hydrogen ion concentration
38	hco3	bicarbonate level
39	hematocrit	hematocrit
40	hemoglobin	hemoglobin
41	platelets	platelets
42	INR	international normalized ratio
43	PT	physical therapy
44	PTT	partial thromboplastin time
45	WBC	white blood cell count
46	RBC	red blood cell count
47	TemperatureC	temperature in Celsius
48	TemperatureF	temperature in Fahrenheit
49	Sodium	blood sodium level
50	Potassium	blood potassium level
51	Chloride	blood chloride level
52	CarbonDioxide	carbon dioxide
53	Glucose	blood glucose level
54	BUN	blood urea nitrogen
55	Creatine	creatine
56	Albumin	albumin
57	TotalProtein	total protein
58	Calcium	calcium
59	Magnesium	magnesium
60	ALT	alanine aminotransferase
61	AST	aspartate aminotransferase
62	AlkPhosphate	alkaline phosphate

continued on the next page

Table A.1 – (*continued*)

Cell	Item	Description
63	TotalBili	total bilirubin
64	DirectBili	direct bilirubin
65	Amylase	salivary amylase
66	CPK	creatinine phosphokinase
67	CPKMB	creatinine kinase-mb
68	Troponin	troponin
69	LDH	lactate dehydrogenase
70	Lipase	lipase
71	UricAcid	uric acid
72	Cholesterol	cholesterol
73	Triglyceride	triglyceride
74	Lactate	lactate
75	PacerRate	pacer rate
76	PreviousWeight	previous weight
77	PreviousWeightF	previous weight F
78	DailyWeight	daily weight
79	IonizedCalcium	ionized calcium
80	epi	epinephrine
81	epi-k	epinephrine-k
82	heparin	heparin
83	insulin	insulin
84	lidocaine	lidocaine
85	propofol	propofol
86	fentanyl	fentanyl
87	morphine	morphine
88	ativan	lorazepam(ativan)
89	vasopressin	vasopressin
90	ditiazem	ditiazem
91	dopamine	dopamine
92	dobutamine	dobutamine
93	levophed	levophed
94	levophed-k	levophed-k
95	amiodorone	amiodorone
96	milrinone	milrinone
97	neo	neo-naclex
98	neo-k	neo-naclex-k
99	nitroprusside	nitroprusside
100	nitro	nitro
101	nitro-k	nitro-k
102	esmolol	esmolol
103	labetolol	labetolol
104	lasix	lasix

continued on the next page

Table A.1 – (*continued*)

Cell	Item	Description
105	aminophylline	aminophylline
106	amrinone	amrinone
107	procainamide	procainamide
108	aggrastat	aggrastat
109	amicar	amicar
110	atracurium	atracurium
111	cisatracurium	cisatracurium
112	doxacurium	doxacurium
113	midazolam	midazolam
114	pancuronium	pancuronium
115	pentobarbitol	pentobarbitol
116	sandostatin	sandostatin
117	reopro	reopro
118	tpa	tissue plasminogen activator
119	vecuronium	vecuronium
120	integrelin	integrelin
121	narcan	narcan
122	fentanyl(conc)	concentrated fentanyl
123	dilaudid	dilaudid
124	precedex	precedex
125	natrecor	natrecor
126	argatroban	argatroban
127	lepirudin	lepirudin
128	nicardipine	nicardipine
129	dopaminedrip	dopamine drip

Appendix B

Deidentified Discharge Summaries of Selected Patient Records

B.1 Discharge Summary for Case Study 1

Table B.1: Discharge Summary for Case Study 1.

```
Admission Date:  2011
Discharge Date:  2011
Date of Birth:
Sex:  F
Service:  Patient passed away on the Cardiothoracic Surgery Service.
HISTORY OF PRESENT ILLNESS: This is a 41-year-old woman transferred from
yyyy Medical Center, where she had been admitted for dyspnea after being
seen numerous times in the congestive obstructive pulmonary disease
exacerbation and worsening hypoxia requiring intubation on August 7th.
She was noted to have elevated CKs and MBs, and therefore was question of
ischemic heart disease versus myocarditis.  She underwent diuresis and had
chest CT scan which showed no evidence of pulmonary embolism or pleural
effusion.  She was further evaluation and possible catheterization.
PAST MEDICAL HISTORY: 1.  Diabetes mellitus.  2.  Borderline high
cholesterol.  3.  History of appendectomy.
MEDICATIONS: Aspirin, Ativan, insulin, Lasix, Heparin, Vasotec, and
dobutamine.
ALLERGIES: Penicillin.
PHYSICAL EXAMINATION ON ADMISSION: Temperature 101.0 F, blood pressure
110/58, pulse 106, respirations 15.
```

continued on the next page

Table B.1 – Discharge Summary for Case Study 1 (*continued*)

Neck is obese with normal carotid upstrokes. Heart normal S1, S2, 2/6 left sternal border systolic murmur. Chest was clear to auscultation bilaterally. Abdomen is soft and nontender, midline scar. Extremities no edema, warm lower extremities. LABORATORIES ON ADMISSION: White blood cell count is 19.6, CK of 554. Chest x-ray: Generous heart size, question reticular nodular densities. Electrocardiogram: Sinus tachycardia at 108, normal axis. Q waves in III and aVF. Borderline left atrial enlargement. Low voltage. HOSPITAL COURSE: Patient was cared for in the CCU, where she was managed for congestive heart failure. She was then taken to the Catheterization Laboratory to evaluate for coronary artery disease and was found to have an 80% ostial stenosis of the left main coronary artery. She became hypotensive while they were attempting to engage the right coronary artery and bradycardic. She was treated with multiple pressors and defibrillation. After she had undergone ventricular fibrillation, Cardiac Surgery was contacted and she was taken to extracorporeal support. She was placed on ECMO. However, she passed away. The family was called and notified.

Protected Health Information in the above patient record has been deidentified.

B.2 Discharge Summary for Case Study 2

Table B.2: Discharge Summary for Case Study 2.

```
Admission Date:  08/26
Discharge Date:  09/05
Date of Birth:
Sex:  M
HISTORY OF PRESENT ILLNESS: The patient is a 24-year-old male with a past
medical history of obesity and obstructive sleep apnea who presented with
complaints of increased swelling and a neck mass times one and half weeks.
The six weeks prior to admission when he was operated on for the removal
of a lipoma.  The operation occurred without incident, and the patient did
well.
Approximately one and a half weeks prior to admission the patient noticed
increased pain and swelling in the lateral aspect of his neck on the left
side.  He presented to the managed conservatively with pain management.
However, the pain and swelling did not resolve.  The patient then began
to experience fevers and difficulty swallowing and difficulty turning
his head.  The patient has not had difficulty breathing.  He denied
trauma, insect bites, or sick contacts.  He presents for evaluation of
his expanding neck mass.
PAST MEDICAL HISTORY: 1.  Obesity.  2.  Obstructive sleep apnea.  3.
Asthma.
MEDICATIONS ON ADMISSION: Home medications include albuterol.
ALLERGIES: There were no known drug allergies.
SOCIAL HISTORY: The patient lives with his girlfriend and works at Boston
Medical Center.  He denied smoking, drug or alcohol use.
FAMILY HISTORY: Family history was remarkable for diabetes mellitus in
multiple first-degree relatives.
PHYSICAL EXAMINATION ON PRESENTATION: Physical examination in the Emergency
Ward revealed vital signs of temperature of 103.9, heart rate 108,
respiratory rate 20, blood pressure 156/60.  In general, the patient
was a morbidly obese African-American male sitting in a chair, breathing
with some effort, but in no apparent distress.  Head, ears, nose, eyes
and throat examination revealed normocephalic and atraumatic.  Pupils
were equal and reactive to light.  Extraocular muscles were intact.  Neck
examination revealed a warm mass posterior to the left ear about 20 cm
X 8 cm long.  Pulmonary examination revealed decreased breath sounds
throughout.  Coronary examination was tachycardic, normal first heart sound
and second heart sound.  No murmurs, rubs or gallops.
```

continued on the next page

Table B.2 – Discharge Summary for Case Study 2 (*continued*)

The abdomen was obese, soft, nontender, and nondistended, with no rebounding, guarding or hepatosplenomegaly, and there were bowel sounds times four. Extremity examination revealed 2+ peripheral pulses. No clubbing, cyanosis or edema. LABORATORY DATA ON PRESENTATION: Admission laboratories were white blood cell count 19.1, hemoglobin 12.7, hematocrit 35.7, platelets 215.. Sodium 131, potassium 4.6, chloride 92, bicarbonate 23, blood urea nitrogen 10, creatinine 0.9. RADIOLOGY/IMAGING: CT scan of the neck revealed a splenius capitus myositis and cellulitis overlying with fluid tracking between plains. There was no evidence of abscess at that time. HOSPITAL COURSE: 1. ENDOCRINE: The patient was newly diagnosed with diabetes mellitus. He was treated with sliding-scales which were adjusted throughout his hospital stay. During his course in the Intensive Care Unit he was maintained on an insulin drip. 2. INFECTIOUS DISEASE: (a) Neck mass: The patient was initially admitted to the general medicine floor for intravenous antibiotics, pain management, and close followup by Otorhinolaryngology. Infectious Disease was consulted for further evaluation of the patient's neck mass, and the consultants recommended the addition of vancomycin, clindamycin, and to continue Unasyn which the patient had been started on. Blood cultures ultimately returned growing coagulase-positive Staphylococcus. The patient's swelling continued to increase, but his respiratory status remained stable. His fevers continued as well. A chest CT examination was checked to examine for extension into the mediastinum, and this was negative. Serial neck CT examinations revealed increase in edema, fat stranding, and possible abscess formation. The patient was then seen on consultation by Neurosurgery to evaluate for extension into the neurologic system. On August 28, the patient was taken to the operating room for drainage of a left posterior neck abscess, and 30 cc to 40 cc of pus was expressed and drained. Extensive necrosis surrounding this are was noted. The patient remained intubated for airway protection following the procedure, and he was transferred in good condition to the Surgical Intensive Care Unit. He was followed by the Medicine consultation team for ongoing diabetes management. Antibiotics were continued and adjusted as required by sensitivities. On August 30, routine laboratories revealed that the patient's hematocrit had been declining, and his white blood cell count had been increasing. Clinically, the patient's neck mass began to increase in size again. That day the patient suffered an episode of hypotension to a systolic blood pressure in the 90s, and he was tachycardic.

continued on the next page

Table B.2 – Discharge Summary for Case Study 2 (*continued*)

He was given large volume intravenous fluids and transfusions of packed red blood cells to restore blood volume and blood pressure. He responded well to this treatment; however, over the course of that evening the patient developed atrial flutter. The patient was felt to be septic and extubation was deferred. Antibiotics were readjusted, and pressors were used p.r.n. to maintain blood pressure. On August 31, the patient's fevers continued. The wound was explored at bedside. There was no new drainage, but there were increased adhesions. White blood cell count continued to rise. The patient was seen in consultation by the Cardiology Service to evaluate for his atrial flutter and to receive a transesophageal echocardiogram to rule out endocarditis. Transesophageal echocardiogram was negative for endocarditis. Cardiology recommended cardioversion and flecainide for stabilization of the patient's rhythm. The patient was transferred to the Medical Intensive Care Unit. His fevers persisted. White blood cell count continued to increase. Hypotension continued to be treated with large volume intravenous fluids to maintain blood pressure. Over the evening of September 1, the patient developed atrial fibrillation to the 130s. He was cardioverted to normal sinus rhythm and extubation was again deferred. The patient's urine also was noted to become rust colored, and his liver function tests began to rise. A repeat CT scan was suspicious for persistent pus in the neck. On September 2, the possibility of meningitis was entertained, and a lumbar puncture was attempted to rule out meningitis. The patient's fevers continued, and the patient developed an increasing oxygen requirement. The patient was again taken to the operating room and an incision an drainage was performed, and a drainage of prevertebral collection of fluid was drained as well. On September 3, the fevers continued. The patient continued to require pressors as necessary to maintain blood pressure. Lumbar puncture was again attempted, and this was again unsuccessful. The patient underwent bronchoscopy to remove mucous plugs felt to be contributing to the patient's increasing oxygen requirements. A repeat CT scan revealed a possible persistent neck abscess and possible lower lung collapse, possibly consistent with acute respiratory distress syndrome. On September 4, the patient was noted to continue to require increasing oxygen. A repeat bronchoscopy was performed and some mucous plugs were obtained. The patient continued to have very high fevers with very elevated creatine kinase levels. The diagnosis of malignant hyperthermia was entertained. The patient's urine output declined, and the possibility of acute tubular necrosis secondary to rhabdomyolysis or sepsis was entertained.

continued on the next page

Table B.2 – Discharge Summary for Case Study 2 (*continued*)

On September 5, the patient was seen on consultation by Nephrology. CVVHD was recommended and undertaken. For the patient's persistent very high fevers dantrolene was given for possible malignant hyperthermia. This resulted in hypothermia. The patient continued to have a worsening metabolic and respiratory acidosis over the day of September 5. THAM and bicarbonate were given to treat for this. Additionally, the patient's platelets were declining, and this was felt to be consistent with disseminated intravascular coagulation. During the day of September 5, the patient continued to have hemodynamic instability. On the morning of September 5, the patient suffered an episode of hypotension and asystolic arrest. Advanced cardiac life support protocols were initiated. The patient was successfully revived; however, remained in very critical condition. During that day, further arrests times two occurred with successful revival but continued hypotension, septic physiology, and bradycardia. Family meetings were held throughout the day to update family members as events unfolded. A late day family meeting took place where the patient's critically ill status was discussed with the family. At the end of that meeting the patient again became bradycardic. At that time the patient's family determined that no further resuscitative efforts should be made. At approximately 6:15 p.m. on September 5, the patient became bradycardic and hypotensive. No intervention was undertaken. The patient became asystolic. The patient was pronounced expired at 18:40 on September 5. An autopsy was performed, per the family's request. DISCHARGE DIAGNOSES: 1. Diabetes mellitus. 2. Staphylococcal sepsis. 3. Neck abscess.

Protected Health Information in the above patient record has been deidentified.

B.3 Discharge Summary for Case Study 3

Table B.3: Discharge Summary for Case Study 3.

```
Admission Date:  2014
Discharge Date:  2014
Date of Birth:
Sex:  M
Service:  Cardiac Surgery
HISTORY OF PRESENT ILLNESS: This is a 66-year-old male who is status post
two aortic surgeries in 199X for dissection; initially in April of and
later in October of .
He had chronic carotid dissection which was followed conservatively, but
over the years had developed a significantly large aneurysm of the aortic
arch measuring 6.8 cm.  He also became symptomatic and started reporting
increased shortness of breath at rest and on exertion as well as increased
fatigue over the past few months.
He underwent a cardiac catheterization as part of his evaluation and was
then referred for this procedure.  He understood the risks involved but
wished to proceed.
PAST MEDICAL HISTORY:
1.  Hypertension.
2.  Hyperlipidemia.
3.  History of atrial fibrillation/atrial flutter; status post three
cardioversions.
4.  History of kidney stones.
5.  History of pneumonia (times two).
6.  History of left hemidiaphragm paralysis secondary to frank nerve damage
during surgery in.
7.  Status post abdominal aortic aneurysm repair in XXX and descending
aortic dissection repair in XXXX.
8.  Status post vocal cord paralysis secondary to vocal cord injury during
surgery in 1991.
9.  Status post left vocal cord Teflon implant in January of 1992.
10.  Diverticulitis.
PHYSICAL EXAMINATION ON PRESENTATION: Examination on admission revealed
the patient's heart rate was 76 (sinus), and his blood pressure was
138/90.  His weight was 200 pounds.  The patient was a well-developed and
well-nourished male in no apparent distress.  The pupils were equal, round,
and reactive to light and accommodation.  The extraocular movements were
intact.  The neck was supple.
```

continued on the next page

Table B.3 – Discharge Summary for Case Study 3 (*continued*)

There was no jugular venous distention. No bruits. The lungs were clear to auscultation bilaterally. No wheezes, rhonchi, or rales. Heart was regular in rate and rhythm. There were no murmurs. The abdomen was obese, soft, nontender, and nondistended. The extremities were warm. There was no edema. No cyanosis. BRIEF SUMMARY OF HOSPITAL COURSE: On ZZZZ the patient was taken to the operating room and underwent a redo sternotomy with arch replacement and innominate vein bypass graft. The operation was complicated by massive bleeding, hypotension, and shock. The patient required multiple blood products. Immediately postoperatively, he developed multiple organ failure including renal failure was oliguric and was started on continuous venovenous hemofiltration, respiratory failure with increased hypoxia and ventilatory support requirement, and liver failure with worsening liver function tests. He remained in cardiogenic shock and required massive anatropic and pressor support to maintain his blood pressure. His chest was left open. During the two days that he spent in the Cardiac Surgery Intensive Care Unit, all organs were supported. He was followed by the Renal Service, Hepatology Service, and the General Surgery Service for question their involvement. Despite extreme attempts to support him his cardiogenic shock persisted, and he became more acidotic with multiple arrhythmias requiring pacing. Over the course of ZZZ, despite attempts at continuous pacing, he became asystolic. The chest was opened, but cardiac function could not be returned. The patient was pronounced dead at 9:05 p.m. The family was contacted and refused autopsy.

Protected Health Information in the above patient record has been deidentified.

Bibliography

[1] Thull B, Popp HJ, Rau G. Man-Machine Interaction in Critical Care Settings. IEEE Engineering in Medicine and Biology 1993;12(4):42–49.

[2] Fackler JC, Kohane IS. Integration of Intermittent Clinical Data with Continuous Data from Bedside Monitors. Children's Hospital's Clinical Data Integration Project Electronic Pages: http://www.chip.org/projects/icuinteg/cbmsfull.html.

[3] Donchin Y, Gopher D, Olin M, Badihi Y, Biesky M, Sprung CL, Pizov R, Cotev S. A look into the Nature and Causes of Human Errors in the Intensive Care Unit. Critical Care Medicine February 1995;23(2):294–300.

[4] Lawless S. Crying Wolf: False Alarms in a Pediatric ICU. Critical Care Medicine 1994;20:981–984.

[5] Cropp AJ, Woods LA, Raney D, Bredel DL. Name that tone. the Proliferation of Alarms in the Intensive Care Unit. Chest 1994;105:1217–1220.

[6] Saeed M, Mark RG. MIMIC-II: A Massive Temporal ICU Patient Database to Support Research in Intelligent Patient Monitoring. Computers in Cardiology 2002;29:641–644.

[7] Mark RG. Integrating Data, Models and Reasoning in Critical Care. National Institute of Biomedical Imaging and Bioengineering Proposal 2003;R01 EB001659.

[8] Abdala OT, Clifford GD, Saeed M, Reisner A, Moody GB, Henry I, Mark RG. The Annotation Station: An Open-source Technology for Annotating Large Biomedical Databases. Computers in Cardiology 2004;31.

[9] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PC, Mark RG, Mietus JE, Moody GB, Peng CK, Stanley HE. Physiobank, Physiotoolkit, and Physionet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 2000 (June 13);101(23):e215–e220. Circulation Electronic Pages: http://circ.ahajournals.org/cgi/content/full/101/23/e215.

[10] Shabot MM. The HP CareVue Clinical Information System. International Journal of Clinical Monitoring and Computing 1997 August;14(3):177–84.

[11] ANSI/IEEE, New York. IEEE Standard for Binary Floating Point Arithmetic, Std 754-1985 edition, 1985.

[12] Moody G. WFDB Applications Guide. Harvard-MIT Division of Health Sciences and Technology, Cambridge, MA, USA, 10 edition, 2005 (June 13). Electronic Edition: http://www.physionet.org/physiotools/wag/wag.htm.

[13] Moody G. WFDB Programmer's Guide. Harvard-MIT Division of Health Sciences and Technology, Cambridge, MA, USA, 10 edition, 2005 (June 13). Electronic Edition: http://www.physionet.org/physiotools/wpg/wpg.htm.

[14] Subversion (SVN) repository for WFDB conversion programs for MIMIC II: http://mimic.mit.edu/svn/wfdb-convert/.

[15] Tansel A, Clifford J, Jajodia S, Segev A, Snodgrass R. Temporal Databases: Theory, Design, and Implementation. Benjamin/Cummings, 1993.

[16] Chomicki J. Temporal Query Languages: A Survey. In Proceedings of the First International Conference on Temporal Logic. 1994; 506–534.

[17] Snodgrass R. The Temporal Query Language TQuel. ACM Transactions on Database Systems 1987 June;12(2):247–298.

[18] Snodgrass R. TSQL2 Temporal Query Language. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[19] Agrawal R, Faloutsos C, Swami A. Efficient Similarity Search in Sequence Databases. In Proceedings of the 4^{th} Conference on Foundations of Data Organization and Algorithms. 1993; 69–84.

[20] Wu Y, Agrawal D, Abbadi A. A Comparison of DFT and DWT based Similarity Search in Time-Series databases. In Proceedings of the 9^{th} International Conference on Information and Knowledge Management. 2000; .

[21] Kanth KV, Agrawal D, Singh A. Dimensionality Reduction for Similarity Searching in Dynamic Databases. In Proceedings of the ACM Special Interest Group on Management of Data (SIGMOD) Conference. 1998; 166–176.

[22] Chan KP, Fu WC. Efficient Time Series Matching by Wavelets. In Proceedings of the 15^{th} International Conference on Data Engineering. 1999; .

[23] Shahabi C, Chung S, Safar M, Hajj G. 2D TSA-Tree: A Wavelet-based Approach to Improve the Efficiency of Multi-Level Spatial Data Mining. In Proceedings of the 13^{th} International Conference on Scientific and Statistical Database Management. 2001; .

[24] Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Journal of Knowledge and Information Systems 2000;.

[25] Keogh E, Chakrabarti K, Mehrotra S, Pazzani M. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. ACM Transactions on Database Systems 2002 June;27(2):188–228.

[26] Saeed M, Mark RG. Efficient Hemodynamic Event Detection Utilizing Relational Databases and Wavelet Analysis. Computers in Cardiology 2001;28:153–156.

[27] Littlefield B. Mastering MATLAB 7. Prentice Hall, 2005.

[28] WFDB_tools Electronic Pages:
http://www.physionet.org/physiotools/matlab/wfdb_tools/.

[29] Douglass M, Clifford GD, Reisner A, Moody GB, Mark RG. Computer Assisted De-identification of free text in the MIMIC-II database. Computers in Cardiology 2004;31.

[30] Michio S, Takahiro Y. Fuzzy-logic-based Approach to Qualitative Modeling. IEEE Transactions on Fuzzy Systems 1993;1(1):7–31.

3233 60